Escuela Politéctica Superior

Dpto. de Ingeniería Informática

Doctorado en Ingeniería Informática y de Telecomunicación

Tesis Doctoral

Genetic Graph-based Clustering applied to Static and Streaming Data Analysis.

Author

HÉCTOR D. MENÉNDEZ BENITO

Advisors

Dr. D. David CAMACHO FERNÁNDEZ
Dr. D. David FERNÁNDEZ BARRERO

A Thesis submitted for the degree of:

Doctor of Philosophy

December 2014

Department: Ingeniería Informática
Escuela Politénica Superior
Universidad Autónoma de Madrid (UAM)
SPAIN

PhD Thesis: "Genetic Graph-based Clustering applied to Static and Streming Data Analysis."

Author: **Héctor D. Menéndez Benito**
Ingeniero en Informática y Licenciado en Matemáticas
Universidad Autónoma de Madrid, Spain

Advisor: **David Camacho Fernández**
Doctor Ingeniero en Informática
(Universidad Autónoma de Madrid)
Universidad Autónoma de Madrid, SPAIN

Co-advisor: **David Fernández Barrero**
Doctor Ingeniero de Telecomunicación
(Universidad de Alcalá de Henares)
Universidad de Alcalá de Henares, SPAIN

Year: 2014

Committee: President:

Secretary:

Vocal 1:

Vocal 2:

Vocal 3:

Dedicado a todos los que han creido en mí

y, en especial, a mis padres por hacerlo posible.

Abstract

Unsupervised Learning Techniques have been widely used in Data Mining over the last few years. These techniques try to identify patterns in a dataset blindly. Clustering is one of the most promising fields in Unsupervised Learning. It consists on grouping the data by similarity. This field has generated several research works which have tried to deal with different problems related to the pattern extraction and data grouping processes. One of the most innovative clustering methodologies is shape-based or continuity-based clustering which tries to group data according to the form they define in the space.

This dissertation is focused on how to apply Genetic Algorithms to the continuity-based clustering problems. Genetic Algorithms have been traditionally used in optimization problems. They are featured by an encoding -which represents the solution space; a population set of chromosomes -which are the potential solutions; and some genetic operations -which are used to evolve the solutions in order to find the best chromosome or solution. The main idea is to take advantage of their potential, generating new algorithms which can improve the performance of classical clustering algorithms, and apply them to static and streaming data.

In order to design these algorithms, this dissertation has been based on the Spectral Clustering algorithm. This algorithm studies the spectrum of a Similarity Graph in order to define the clusters. The clusters defined by Spectral Clustering usually respect the data continuity. Using this idea as a starting point, different graph-based genetic algorithms have been designed to deal with the continuity-based clustering problem. The different algorithms developed have been divided in three generations:

- The first generation is based on genetic graph-based clustering algorithms. In this generation we combined graph-based clustering and genetic algorithms to generate a graph topology among the data, in order to find the best way to cut the graph. This cutting process is used to discriminate the final clusters. The main idea is to use hybrid algorithms which combine different metrics extracted from graph theory. In order to evaluate the performance on real-world problems, these algorithms have been also applied to text summarization.
- The second generation is based on multi-objective genetic graph-based clustering algorithms. This generation introduces the Pareto Front generated by the different fitness functions used in the genetic search. The Pareto Front is used to study the solution space and provides more robust and accurate solutions. During this generation we also used co-evolutionary algorithms to include the number of clusters in the search space.
- Finally, the last generation is focused on large and streaming data analysis. During this generation the previous algorithms have been adapted to deal with

large data, combining different methodologies such as online clustering and MapReduce. The main idea is to study their performance compared with other algorithms.

The dissertation also includes a description of other graph-based bio-inspired algorithms, in this case Ant Colony Optimization Clustering algorithms, which have been designed during the dissertation, in order to extend the range of study to other bio-inspired areas.

Finally, with the purpose of evaluating the algorithms of the different generations, we have compared them with relevant and well-known clustering algorithms using synthetic and real-world datasets extracted from the literature and the UCI Machine Learning Repository.

Resumen

Las técnicas de aprendizaje no supervisado han sido ampliamente utilizadas en minería de datos en los últimos años. Estas técnicas tratan de extraer patrones de un conjunto de datos de forma ciega. Dentro de las mismas, el Clustering es uno de los campos más prometedores. Éste consiste en la agrupación de los datos por similitud. Este campo ha generado varios trabajos de investigación que han tratado de hacer frente a diferentes problemas relacionados con la extracción de patrones y los procesos de agrupación de datos. Una de las metodologías de clustering más innovadoras se basa en agrupar los datos por continuidad, respetando la forma que éstos definen en espacio en el que se encuentran.

Esta tesis se centra en la manera de aplicar algoritmos genéticos a los problemas de clustering basado en continuidad. Los algoritmos genéticos han sido utilizados tradicionalmente en problemas de optimización. Se caracterizan por una codificación -que representa el espacio de soluciones-, una población o conjunto de cromosomas -que son las soluciones potenciales dentro de este espacio-, y algunas operaciones genéticas -que se utilizan para evolucionar las soluciones con el fin de encontrar el mejor cromosoma o solución-. La idea principal es aprovechar el pontencial de los algoritmos genéticos generando nuevos algoritmos que pueden mejorar el rendimiento de los algoritmos clásicos aplicados tanto a datos estáticos como a flujos continuos de datos.

De cara a diseñar estos algoritmos, esta tesis doctoral utiliza el algoritmo de Spectral Clustering como punto de partida. Este algoritmo estudia el espectro de un grafo de similitud con el fin de definir las agrupaciones o clusters. Los grupos definidos por Spectral Clustering suelen respetar la continuidad de los datos. Utilizando esta idea, se han diseñado diferentes algoritmos genéticos basados en grafos para hacer frente al problema de agrupación basada en continuidad. Los diferentes algoritmos desarrollados se han dividido en tres generaciones:

- La primera generación se basa en algoritmos de clustering genéticos basados en grafos. En esta generación se han combinado técnicas de Graph Clustering y algoritmos genéticos para generar una topología de grafo entre los datos, con el fin de encontrar la mejor manera de cortar el grafo. Este proceso de corte se utiliza para discriminar los grupos finales. La idea principal es utilizar algoritmos híbridos que combinan diferentes métricas extraídas de teoría de grafos. Con el fin de evaluar el comportamiento de los algoritmos en problemas del mundo real, estos algoritmos se han aplicado al problema de cómo generar resúmenes automáticos.
- La segunda generación se basa en algoritmos multi-objetivo de clustering genético basado en grafos. Esta generación introduce el Frente de Pareto, generado por las diferentes funciones de fitness utilizadas en la búsqueda genética. El frente de Pareto se utiliza para estudiar el espacio de soluciones y proporcionar soluciones más robustas y precisas. Durante esta generación también utilizamos algoritmos co-evolutivos de cara a incluir el número de clusters en el espacio de búsqueda.

- Finalmente, la última generación se centra en el análisis de grandes cantidades y flujos de datos. Durante esta generación los algoritmos anteriormente mencionados se han adaptado para hacer frente a grandes volúmenes de datos, combinando diferentes metodologías como el clustering online y MapReduce. La idea principal es estudiar su rendimiento en comparación con otros algoritmos.

La tesis también incluye aportaciones de otros algoritmos bio-inspirados basados en grafos, en este caso, algoritmos de clustering usando optimización por colonias de hormigas. Estos algoritmos han sido diseñados durante el desarrollo de la tesis para ampliar el rango de estudio a otros entornos bio-inspirados.

Por último, con el fin de evaluar los algoritmos de las diferentes generaciones, se han comparado con algoritmos de clustering conocidos. El rendimiento de estos algoritmos se ha medido utilizando conjuntos de datos sintéticos y reales extraídos de la literatura y del repositorio UCI de Machine Learning.

Agradecimientos

"Las paredes siguen en su sitio hoy,
posiblemente lo único que no cambió
porque ni este es mi cuarto
ni esta soy yo..."

- María Peláez (Alsondelpez)

Aunque es difícil poder decir con claridad y precisión a quién se puede agradecer este libro, es quizás más fácil hablar a quién se puede agradecer que me haya aguantado hasta aquí. Lo primero, me gustaría dar las gracias a mis compañeros de AIDA, especialmente a Víctor, Cristian, Fernando y Fran que, incluso en los momentos duros, han querido seguir con una sonrisa en la cara. También a mi compañero Antonio por echarme una mano con todos los temas de papeleos.

Sin ser del todo del grupo, también quiero agradecer a mis estudiantes: Carlos, Miguel y Rafa, su motivación y esfuerzo a la hora de trabajar, su madurez para afrontar el futuro y su templanza ante la envidia de aquellos que no han sido capaces de valorarles.

Dentro de este marco, me gustaría agradecer a mis colaboradores su trabajo durante todo este tiempo, en especial, a Laura Plaza y a Shintaro Okazaki que me han podido mostrar nuevas perspectivas de trabajo. Además, me gustaría agradecer a Fernando Otero que me acogiera en la Universidad de Kent y que me enseñara tanto algoritmos de hormigas como a hacer análisis más rigurosos. During my research stay in Kent, I also would like to thank Kristopher Welsh, Caroline Li and Janine Jarvis, their kindness to make me feel part of the university.

Siguiendo con el entorno institucional, quiero agradecer a la Universidad Autónoma de Madrid la oportunidad de realizar esta tesis doctoral con financiación en los tiempos que corren. Aparte, a todas las personas que trabajan tanto en la administración de la Escuela Politécnica Superior como en la Secretaría del Departamento de Informática y en el servicio de Investigación de Rectorado, por su buena disposición para ayudarme con todos los trámites burocráticos. Dentro de las personas que he conocido en los 10 años que llevo en esta institución, quiero aprovechar para agradecer también a ciertas personas más específicas el haberme tratado como parte activa de la universidad. Estas personas son: Isabel Castro, Rosa Carro, Álvaro Ortigosa y Antonio Álvarez-Ossorio.

Dentro de mi marco formativo, me gustaría agradecer a muchos de mis profesores su ayuda a la hora de formarme, pero sobre todo a Ana Bravo, que me enseñó a pulir un buen trabajo y a trabajar de una forma estructurada y constante.

Una de las principales personas a las que quiero agradecer esta tesis es a mi director, David Camacho, que no sólo me ha dirigido la tesis, si no que, además, ha llevado mi carrera en todos estos años. Gracias a sus consejos, he pude obtener una beca FPI y, más adelante, mi plaza de profesor y, además, he podido tener contacto con muchos proyectos y con muchos colaboradores que, en mayor o menor medida, me han enseñado a avanzar. Aparte, me gustaría agradecerle su guía, su confianza y el haber cogido todo el peso de la tesis.

Dentro de un marco más personal, me gustaría agradecer a mi compañera, Gema Bello Orgaz, todos los ratos que me ha tenido que soportar y, sobre todo, que viniese conmigo de estancia a Kent. Aparte, también quiero agradecerle sus consejos a la hora de ayudarme a estructurar partes de mi trabajo y su buena predisposición en todas las colaboraciones que hemos realizado juntos, desde que empezamos como compañeros de prácticas del máster en la asignatura de Computación Evolutiva. También, en el mismo contexto a mi antiguo compañero de prácticas y amigo, Saúl Vargas, por haberme abierto los ojos ante muchos de mis errores y por haberme enseñado a utilizar las herramientas adecuadas para cada cosa (o a buscarlas). Además, por haber sido un buen amigo que siempre ha sabido apoyarme en momentos complicados y que ha sabido entender los problemas de forma muy parecida a la mía.

Dentro del plano personal, me gustaría agradecer a mis amigos el haber estado siempre ahí para apoyarme en los momentos más difíciles, en especial, a Alberto, Ricardo, Alba, Marina y Ferrero que siguen ahí después de tanto tiempo. También a los nuevos amigos: Antonio, las dos Sara, Juan y, con mucho carioña, a Mariola, cada rato compartido. De la universidad, me gustaría agradecer a mis amigos de la Siglo XXI todos los momentos que hemos pasado juntos, en especial, a Karim, a los dos Alberto, Pedro, Ñoto, Jorge, Eva, Guille, Pedro J., a las dos Paloma, Manolo y Luis Jorge. Con ellos he podido disfrutar y aprender muchas cosas de lo compleja que puede ser la vida. También del Club de Robótica e Informática a Irene, Víctor, Lucas, Carlos, Álvaro y Antonio, por haberme ayudado a mantener las asociaciones cuando tuve que dejarlas por la tesis. Por otro lado, a mis amigos de la Erasmus: Silvia, Santi, Jesús, Inés, Coke, Raquel y María, por ser como mi familia durante todo un año. De forma muy especial, quiero agradecer a M^{a} Carmen su cariño y sus ánimos en los momentos más complicados, a Elisa todas las veces que me ha tenido que escuchar y todos los consejos que me ha dado, tanto personales como profesionales y a Deza el haber sido un apoyo importante siempre que las cosas se torcían y mi mejor consejero. Por otro lado, me gustaría agradecer a Luismi todo su apoyo durante estos 10 años, desde que entré en la universidad y éramos compañeros hasta ahora. Con especial aprecio, me gustaría agradecer a José Luis todo lo que ha tenido que soportarme y todos sus esfuerzos por animarme siempre que he pasado por momentos muy difíciles, además de todos los conciertos de los que hemos disfrutado, que, muchas veces, han sido una vía de escape necesaria para reconciliarme con el mundo.

A mi familia quiero agradecerle todo, desde mis abuelos, Pedro y Manolo, que plantaron los cimientos, a mis abuelas, Consuelo y Dolores que hicieron todo lo que estuvo en sus manos para sacarlo adelante y trasmitir su cariño en cada momento. A mis primos: Pedro, María, Teresa, Esther, Raquel, Rosa e Irene, su compañía en estos años y a mis tíos: Manuel, Bernardo, Geni, Pilar, Esperanza y Rafa, su paciencia y

sus consejos en muchos marcos. También me gustaría agradecer a mi familia política su apoyo, en especial, a Pepi y Pedro, mis suegros, a Elena, mi cuñada y a Jesús y Rebeca, mis primos políticos, por haberme acogido como uno más dentro de la familia. También, quiero agradecerle a Álvaro cada minuto juntos de estos últimos tres años, todo su cariño y su amor, sus consejos, su apoyo, su paciencia y sus abrazos, que me han dado muchas fuerzas para poder alcanzar esta meta y que se han convertido en un punto de partida para un futuro muy próspero juntos.

Finalmente, quiero agradecer, en especial, a mis padres Cándido y M^{a} Carmen y a mi hermana Carmen su infinita paciencia, sus forma de contagiarme cada día las ganas de luchar y de superarme a mi mismo y su confianza, que han conseguido que haya podido dar todos y cada uno de los pasos que me han llevado hasta este preciso momento y que en un futuro, aún pueden llevarme muchos más lejos.

A todos vosotros y a todos los que habéis estado ahí, muchas gracias.

Contents

List of Figures

List of Tables

List of Acronyms and Symbols

$\mathbb{R}$	Set of Real Numbers
$\mathbb{R}^n$	n-dimensional vector space for Real Vectors
$G(E, V)$	Graph with V a set of vertex and E a set of edges
$sim()$	Similarity Function
X	Dataset with n instances, where $X = \{x_1, \ldots, x_n\}$
C	Set of Clusters with k clusters $C = \{c_1, \ldots, c_k\}$
EM	Expectation Maximization
SC	Spectral Clustering
GGC	Genetic Graph-based Clustering Algorithm
MOGGC	Multi-Objective Graph-based Clustering Algorithm
CEMOG	Co-Evolutionary Multi-Objective Genetic Graph-based Clustering Algorithm
MOGGSC	Multi-Objective Graph-based Streaming Clustering Algorithm
MOGGOC	Multi-Objective Graph-based Online Clustering Algorithm
L	Laplacian Matrix
L_{sym}	Laplacian Similarity Matrix
W	Weighted Matrix
A	Adjacency Matrix
KNN	K-Nearest Neighbourhood
MinCut	Minimal Cut
NCut	Normalized Cut
GTC	Genetic Text Clustering Algorithm
SACOC	Spectral-based Ant Colony Optimization Clustering Algorithm
MACOC	Medoid-based Ant Colony Optimization Clustering Algorithm
CC	Clustering Coefficient
WCC	Weight Clustering Coefficient
AD	Average Distance
GA	Genetic Algorithm
MOGA	Multi-Objetive Genetic Algorithm
ACO	Ant Colony Optimization

Chapter 1

Introduction

"Como dijo Jack el Destripador: 'Vamos por partes...' "

- David y José Manuel Muñoz (Estopa)

This chapter motivates and overviews this dissertation. Firstly, we briefly introduce Clustering as the problem focused. Then, Section 1.2 motivates the questions that are addressed later. After that, in Section 1.3, Graph Clustering and Genetic Algorithms are briefly introduced in order to provide a basic framework to state the research questions, that are reported in Section 1.4. Next, the dissertation structure is described in Section 1.5 and, finally, the main contributions and the associated publications are described.

1.1 The Clustering Problem

Machine Learning is one of the most important fields in Artificial Intelligence. The different learning processes have been widely studied in several environments where the most remarkable are data analysis and statistical inference. Currently, statistical methods have been extended, and there are new methods based on bio-inspired approaches. These methods are usually based on Evolutionary Computation or Swarm Intelligence methods. The former is based on the evolution of chromosomes in a population, whereas the latter is based on the emergent behaviour of small agents which collaborate in order to complete a task.

Machine Learning has several subfields, each of them assigned to a learning methodology which its own constrains. The most relevant are [5]: Supervised Learning, Unsupervised Learning, Reinforcement Learning and Game Theory Learning. The first methodology is focused on those problems whose goal is to extract a general model or patterns having information about the quality of the model. The second is focused on finding a way to extract patterns from the data blindly. The third tries to give some information during the process through a reward or punish. Finally, the last is a generalization of Reinforcement Learning where several agents are used. This work has been focused on unsupervised techniques.

The unsupervised learning methods are mainly based on clustering techniques [40]. These techniques were designed to find hidden information, or patterns, in a dataset grouping the data with similar properties in clusters. Clustering is a large field and comprises several methodologies. If we consider a dataset $X = \{x_1, \ldots, x_n\}$ where x_i is a data instance and n is the

number of instances, the algorithm goal is to generate k sets called clusters. We can denote $C = \{c_1, \ldots, c_k\}$ as the set of all clusters and c_i as a cluster. The different methods to group data in clusters can be divided in three main categories [90]:

- **Partitional** [126]: Consists of a disjoint division of the data where each element belongs only to a single cluster. In this case, we have the following restrictions: $c_i \cap c_j = \emptyset$, $c_i \neq \emptyset$ and $X = c_1 \cup \cdots \cup c_k$.
- **Overlapping** [27] (or non-exclusive): Allows each element to belong to multiple clusters or none cluster. In this case, there are less restrictions, and the intersection of two clusters is not necessarily empty. However, each cluster should have at least one element.
- **Hierarchical** [109]: Nests the clusters formed through a partitional clustering method creating bigger partitions, grouping the clusters by hierarchical levels. In this case, if cluster c_i is below cluster c_j and they are hierarchically connected, then $c_i \subsetneq c_j$.

The thesis has been focused on the first category: partitional clustering. In order to deal with the partitional problem, it is important to consider the model which is used to extract the patterns from the data. Usually, classical statistical models use an estimator (e.g., a Gaussian distribution) whose parameters are adapted in order to fit with the data distribution. However, modern techniques try to avoid the usage of estimators in order to generalize the problem. The main methodologies are usually categorized as [40]:

- **Parametric or Model-based clustering** [40]: consists of an estimator based on a mixture of probabilities whose parameters are estimated. This estimation fixes the model to the dataset.
- **Non-Parametric clustering** [193]: there is not an initial probability model or estimator.
- **Semiparametric method** [40]: a combination of parametric and non-parametric methods.

This work has been focused on Non-Parametric Partitional Clustering. One of the most classical non-parametric approaches is the Spectral Clustering algorithm introduced by Ng et al. [155] in 2001. This algorithm –which is the stating point of this dissertation; is based on the process to cut a similarity graph in order to separate the data (represented as that graph). The methodology is usually divided in three main steps:

1. A similarity function is applied over an original dataset to construct a Similarity Graph among the information.
2. The Spectrum of the Similarity Graph is calculated in order to generate a projective space through its eigenvectors.
3. K-means (or other partitional clustering technique) is applied to the matrix formed by the k-first eigenvectors to discriminate the information and assign the final clusters.

Using this methodology the algorithm is able to separate the data keeping information about the continuity in the data instances. This is usually called a shape-based, or continuity-based, separation. The algorithm is not just considering where are the centres of the clusters, it tries to discover the form generated by the data in the search space and tries to discriminate different forms. As it will be explained in the following section the algorithm have some problems according to the robustness which are the main starting point of this dissertation.

1.2 Motivation of the dissertation

The main problem related to Spectral Clustering is its sensitivity to the definition of the similarity function. The algorithm is generally not too robust, it usually converges to local solutions when noisy information appears, as Chang and Yeung exposed in [37]. In order to fix this problem, some studies were based on the improvements of the parameters selection for the similarity function [37]. Other solutions are focused on the selection of the partitional clustering algorithm for the third step of SC [196]. This work has designed new algorithms based on Genetic Algorithms (GA) to improve the robustness of the clusters selection taking the Similarity Graph as a starting point. The new algorithms look for new solutions that could aid to manage this problem.

Genetic algorithms [45, 187] provide a framework to find solutions which are more accurate for a particular heuristic. The chromosomes of the population are usually solutions and the algorithm tries to find the best individual according to a fitness function. However, these individuals are not always the best solution. The search space generated by the heuristic should be smooth enough to find a good path to the best solution -i.e. a minimum or a maximum of the heuristic; but there are several cases where the search space contains local minimum which are a trend of the algorithm convergence path, producing premature convergence. Figure 1.1 shows an example of a search space with local solutions and global solutions. In this case, the search space is smooth, however, there are some problems where it is discrete or contains holes. Due to this problem, it is not only important to find a good evaluation heuristic, it is also important to define a smooth search space or to provide a good Genetic Algorithm which can deal with local solutions.

When Spectral Clustering deals with the non-parametric problem, it generates a Similarity Graph among the data providing a new topology. This graph is cut by the algorithm choosing the best data separation. However, the eigenvector decomposition is usually computationally expensive. Due to the graph topology provides the shape or continuity information, we have considered to keep the graph and apply the genetic approaches to it. Then, the graph becomes the search space because we do not have information about the original space, and we are not able to use it. This search space is discrete and is not usually smooth so we need to take care about the Genetic Algorithm and the heuristic. This has motivated the research in different ways, such as Multi-Objective algorithms [133] and Co-Evolutionary algorithms [138].

Other important problem in clustering is to define the number of clusters, which is a difficult task. Several clustering algorithms consider that the human must introduce the number of clusters and they will optimize these clusters according to a metric or cost function. There are some algorithms which deal with this problem. In this dissertation, we also deal with this issue generating a Co-Evolutionary approach where different populations try to deal with the number

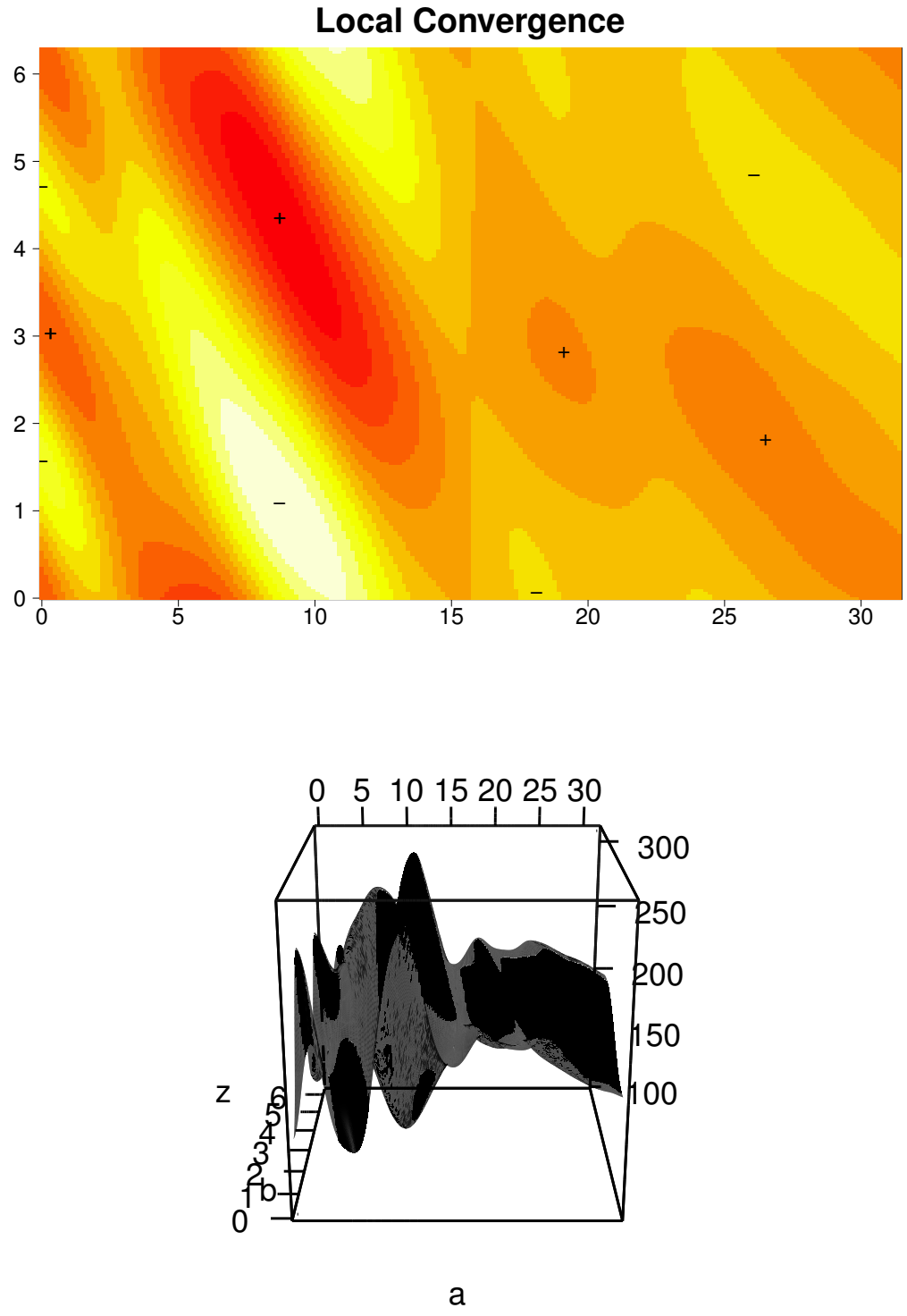


Figure 1.1: Example of a function with a local minimum and maximum.

of clusters.

Data structure is also an important issue. The first step in data analysis is to understand the data, in order to discover patterns within it. Here, we face two kind of data: static data, which can be usually taken offline and is fixed during the analysis, and streaming data which are usually large data quantities and could be constantly appearing. The former is the most frequent, but there are several data representation such as images or texts that needs a special attention. The later is a consequence of current communications, and there are two main ways to deal with it: large data algorithms or online algorithms. Here, we adapt our approaches to both kinds of data.

Apart from the theoretical approaches presented above, this dissertation also pretends to introduce a practical problem which has been studied using genetic graph-based algorithms. It is related to documents summarization which is a static data analysis problem.

Finally, it is also important to understand how these graph-based clustering approaches can be useful in different bio-inspired domains applied to the clustering problems. For this reason, the dissertation concludes with some approaches based on Ant Colony Optimization (ACO), a bio-inspired graph-based technique extracted from Swarm Intelligence area which imitates the foraging behaviour of the ants in order to find the best path between the nest and the food source.

1.3 Problem statement

The graph analysis is the main problem which this dissertation faces. The first idea for graph analysis is based on network analysis [54]. Given a network which is represented as a graph, classical techniques analyse the different properties of the network through various measures. The principal measures are the Clustering Coefficient [54] (CC), and the Average Distance (AD) of the elements. There are several variations of them, such as the Weighted Clustering Coefficient (WCC) [20] which considers weights in the edges of the graph. In order to support a good election for the heuristics, some measures will be analysed in detail in the next sections.

One of the main problems of AD or other Complex Network measures is that they do not consider the continuity of the dataset which is important in graph-based clustering approaches. The continuity is the "form" defined by the data, for example, an object form into an image. Therefore, other different heuristics based on well-known algorithms (such as K-Nearest Neighbours [48] or Minimal Graph Cut [177]) have also been tested and combined to improve the results through Genetic Algorithms (GA).

Genetic algorithms have been traditionally used in optimization problems. The complexity of the algorithm depends on the codification and the operations that are used to reproduce, cross, mutate and select the different individuals (chromosomes) of the population [45, 187]. These kinds of algorithms apply a fitness function which guides the search to find the best individual of the population.

Two essential elements in any GA design are the fitness function and the encoding of the individuals. The first one is used to guide the evolutionary search [45, 187], the second one determines the search space and landscape [175].

In this particular case of clustering, the encoding has a convergence problem. The literature has addressed this issue and there are many approximations to the encoding of clustering in GA, (see [90] for a complete analysis of this problem). Given the importance of this topic to the success of the GA, we start this work studying two possible encodings, comparing their main features and subsequent performance.

Clustering is essentially a search problem of a function that maps data into clusters, just the type of problem that EAs handle well. The first algorithm introduced in this dissertation is called Genetic Graph-based Clustering algorithm (GGC). GGC has a fitness function based on a metric to measure the cluster quality. It is inspired by Complex System Analysis and Graph Theory [54, 151, 200] and specially focused on K-Nearest Neighbours [48] and Minimal Graph Cut [177], as was mentioned above.

In order to assess the performance of GGC, we have carried out a collection of experiments that compare GGC performance and robustness to those of some classical algorithms, like K-means, EM and, of course, SC. Besides, it has been applied to automatic text summarization. The summarization methodology used was introduced by Plaza et al. in [162]. This technique is based on several steps, but the main step -which discriminate the concepts; is based on a clustering algorithm. The original algorithm uses a centroid based approach, however, the main idea of this work is to apply a continuity-based approaches which can compete with the classical summarization and commercial process. This methodology has been applied to summarization of biomedical documents.

GGC uses the same graph representation that SC and also improves the robustness of the clustering results related to the metric used to measure the data similarity. However, this algorithm has the same memory usage problems than SC: It generates a matrix comparing all data instances pair to pair, whether the problem is focused on large datasets, this matrix becomes extremely big and it is difficult to store (and therefore to compute) all its information.

After the introduction of GGC, we propose a new algorithm named Multi-Objective Genetic Graph-based Clustering Algorithm (MOGGC). It is based on GGC and combines Multi-Objective Genetic Algorithms (MOGA) [42] with graph-continuity metrics to achieve two goals: Lower memory consumption and increased solution quality in comparison to GGC. In order to assess MOGGC performance, we compare it against the three classical clustering algorithms (K-means, EM and SC) and the original GGC. The experimentation carried out involves synthetic and well-known UCI datasets.

In order to get over the fixed number of clusters issue, a new algorithm called Co-Evolutionary Multi-Objective Genetic Graph-based Clustering (CEMOG) algorithm is introduced. The contribution of CEMOG is the development of a new partitional clustering algorithm that solves the k-determination problem of MOGGC. To this end, CEMOG uses co-evolution to simulate macro and micro evolution in the Genetic Algorithm. In this way, the value of k is introduced in the evolutionary search and eventually the Pareto front provide a set of k corresponding to the trade-off of the solutions.

In order to deal with stream data, we have adapted the algorithm to face large data. Stream data is usually the information provided by real-time systems, such as mobile companies, sensors, etc. Usually, the data defined as stream data is large and the algorithms need some scalability effort to deal with them. There is another important approach for stream data which is related to the data flow. Normally, these data come in a specific order, which means that the algorithm

has to process the data continuously.

The first approach can be identified as a large data problem, the algorithm only has to deal with a large dataset which can be faced with approaches such as Map Reduce [51]. In this work we adapt the original algorithm MOGGC to process large data problems, reducing the search space through a 2-step clustering algorithm. The first step reduces the search space through a clustering algorithm such as K-means. The second step takes the basic information from the first step, in order to generate more complex clusters.

The second approach is an online problem, in this case it is necessary to understand how online problems works. First, each data can usually be processed once (usually online approaches relax this restriction, but it is the theoretical restriction), and the algorithm might not keep information about the data instances. This produces several limitations, the main problem is that we need a concrete point as an input data, which implies a featured search space. The algorithm can define the solutions in the search space, keeping only information about centroids. To deal with this problem, we have adapted the first approach. We need a featured search space, hence, we are only able to use a centroid algorithm. Apart of this, we need to put an online centroid based algorithm. With this information we can adapt the original algorithm to an online approach in order to deal with the data streams.

Finally, it is important to analyse and study other relevant bio-inspired approaches which could be used to improve our clustering approaches. In this case, we have focused our attention on Ant Colony Optimization (ACO). Ants usually tries to find an optimal path, so it is necessary to adapt the clustering problem to a path optimization problem.

The first approach for ACO Clustering is based on the design of a medoid-based approach. This approach tries to generalize a previous centroid-based ACO Clustering algorithm including the characteristic such as that the features of the search space is not necessary known. This approach, even if is not totally connected with the continuity-based problem, provides a new graph-based approach to solve clustering problems.

The second approach is an "spectralization" of the ACOC algorithm, this approach applies the same principles which are applied to Spectral Clustering, however we changed the clustering algorithm which separates the data trying to obtain better results than the original Spectral Clustering.

From this problem and research statement, this PhD dissertation will try to provide some insights to several research questions that will be describe in the next section.

1.4 Research Questions

The main *research question* that is faced in this PhD Thesis can be described as follows:

Spectral Clustering has been extensively used in Clustering Applications. This kind of algorithms generates a graph topology over the data, but it has several problems according to its robustness when it discriminates the groups. Taking the same graph topology, the main question is whether it is possible to optimally generate another clustering algorithm, oriented to cut the graph, through genetic algorithms. Also it is important to study different

application fields and data structures to check its performance. Based on this idea, it is also important to analyse other bio-inspired approaches based on graph representations.

Previous general research question can be decomposed into more specific research questions:

- **Q1:** Is it possible to improve the results of classical algorithms -specially Spectral Clustering, using genetic graph-based approaches?
- **Q2:** How these approaches deal with the robustness problem of classical Spectral Clustering algorithm?
- **Q3:** How can genetic algorithms and graph-based structures be combined to create these algorithms?
- **Q4:** How modern genetic-based methodologies can help to improve the quality of the algorithms?
- **Q5:** How can these algorithms deal with static data?
- **Q6:** How can these algorithms deal with streaming data?
- **Q7:** Is there any other bio-inspired methodology promising for graph-based approaches?

Based on the specific research questions, we can state the main goal:

To create genetic graph-based clustering algorithms and heuristics to improve the robustness and solutions of classical partitional clustering algorithms, and to study its application to different problem domains.

1.5 Structure of the thesis

The dissertation has been structured in seven chapters. Following, a brief description of the chapter contents are given:

- **Chapter 1: Introduction.** It provides a general background, context and motivations. The main objectives and research questions are stated, as well as the dissertation structure, main contributions and publications.
- **Chapter 2: State of the Art.** It introduces the State of the Art around Clustering and Genetic Algorithms. This chapter contextualizes the different contributions of the PhD thesis.
- **Chapter 3: The Genetic Graph-based Clustering Approaches.** This chapter introduces the main problems around the Spectral Clustering algorithm and present the first methods developed to deal with these problems from different perspectives. Here, the analysis is focused on the encoding, operations and different fitness functions which can deal with the clustering problem. Also, the algorithm presented is applied to the Automatic Summarization field, in order to evaluate its performance in real applications.

- **Chapter 4: Multi-Objective Genetic Graph-based Approaches.** It provides some evolution in the context of genetic algorithms applied to clustering. This chapter tries to show how the GGC algorithm can be improved and the accuracy of these improvements. The main improvements are based on memory optimization and the automatically selection of the number of clusters.

- **Chapter 5: Genetic Graph-based Clustering for Streaming Data Analysis.** This chapter is focused on the data stream problem and how we can design an implementation based on the previous algorithms to face it. Here, we introduce two new algorithms to deal with large data (Massive Data and online problems).

- **Chapter 6: Other Bio-inpired Approaches.** During the PhD thesis other bio-inspired graph-based approaches have been designed and tested. These ideas are based on Swarn Intelligence, more specifically Ant Colony Optimization. In this chapter, new algorithms developed during this thesis are presented around the ACO Clustering problem.

- **Chapter 7: Conclusions.** The Research Questions described in Chapter 1 are addressed in order to provide some answers, based on the results obtained from this research.

1.6 Publications and Contributions

This chapter explains the contributions which have been generated by this thesis. We describe the contributions by chapter which have been developed during the thesis dissertation:

- **Chapter 2**: This chapter is focused on all the preliminary research which has been helpful for this dissertation. It also includes some fields where the clustering approaches are usefully applied and some examples about the different applications which have been done during this work in the different fields. During this thesis, we have been applied the clustering problem to Robossocer [97], Feature selection [134], Soccer [135], Eurovision Contest [24], Community finding problems [22], Baseball forecasting [145], Twitter [156] and Image Segmentation [136].

  Journals:

  - G. Bello-Orgaz, H. D. Menéndez, and D. Camacho. Adaptive k-means algorithm for overlapped graph clustering. *International Journal of Neural Systems*, 22(05):1250018, 2012. **Impact Factor (2012): 5.054. Q1**

- **Chapter 3**: The main contributions of this chapter are related to the GGC algorithm. The first contribution presents the algorithm [140] providing an overview of how it performs. The second contribution is a deep study of GGC trying to remark its robustness and its advantages compared to Spectral Clustering [139]. The second contribution of this chapter are based on the different applications for the genetic algorithms to the summarization problem. The summarization problem has shown good results using the original GGC algorithm [143], however, an adaptation of this algorithm shows that considering some text-based metrics the results can be improved [144].

  Journals:

  - H. D. Menéndez, D. F. Barrero, and D. Camacho. A genetic graph-based approach for partitional clustering. *Int. J. Neural Syst.*, 24(3), 2014. **Impact Factor (2013): 6.056. Q1**.

  Indexed Conferences:

  - H. D. Menéndez, L. Plaza, and D. Camacho. Combining graph connectivity and genetic clustering to improve biomedical summarization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2740–2747. IEEE, 2014. **Core A**.

  - H. D. Menéndez and D. Camacho. A genetic graph-based clustering algorithm. In H. Yin, J. Costa, and G. Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, volume 7435 of *Lecture Notes in Computer Science*, pages 216–225. Springer Berlin / Heidelberg, 2012. **Core C**.

- **Chapter 4**: The principal contributions of this chapter are the two algorithms which are presented. The first algorithm is MOGGC [133] which is a multi-objective approach that optimize the memory usage compared with the original GGC. It also modifies the metrics used in the search space. The second algorithm is CEMOG [138], which is an evolution of MOGGC. This algorithm tries to face the number of clusters problem. It presents an approach where the algorithm is also able to adapt the k parameter in order to fix the number of cluster.

  Indexed Conferences:

  - H. Menéndez, D. F. Barrero, and D. Camacho. A multi-objective genetic graph-based clustering algorithm with memory optimization. In *2013 IEEE Conference on Evolutionary Computation*, volume 1, pages 3174–3181, June 20-23 2013. **Core A**.
  - H. D. Menéndez, D. F. Barrero, and D. Camacho. A co-evolutionary multi-objective approach for a k-adaptive graph-based clustering algorithm. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2724–2731. IEEE, 2014. **Core A**.

- **Chapter 5**: The contributions which are presented in this chapter are according to both problems: how to deal with large data and how to deal with large data in an online way.

- **Chapter 6**: This last chapter shows the contribution in the ACO Clustering problem. The first contribution is a Medoid-based clustering algorithm called MACOC [137]. The second is a Spectral Approach to the problem called SACOC [142].

  Indexed Conferences:

  - H. Menéndez, F. E. B. Otero, and D. Camacho. Macoc: a medoid-based aco clustering algorithm. In *Swarm Intelligence - 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings*, pages 122–133. Springer, 2014. **Core B**.

  Other Conferences:

  - H. D. Menéndez, F. E. Otero, and D. Camacho. Sacoc: A spectral-based aco clustering algorithm. In *Intelligent Distributed Computing VIII*, pages 185–194. Springer, 2015

Chapter 2

State of the Art

"I don't know anything, but I do know that everything is interesting if you go into it deeply enough."

- Richard Feynman

This section starts with a general introduction to Data Mining and Machine Learning areas. After this brief introduction, it presents the different kinds of data which are usually found in Data Mining and how this data is preprocessed and normalized. Next, it is focused on the clustering methods, introducing classical clustering techniques, graph-based techniques, genetic-based techniques and stream data analysis techniques. After the methodology description, several applications are shown, specially focused on those applications which have been carried out at the same time to this thesis project.

2.1 Data Mining

Data Mining is "the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques" [111]. The Data Mining techniques can be divided in 5 main steps:

1. **Data Extraction:** The data extraction problem consists on obtaining the datasets which will be analysed. There are several public databases, for example, those which are contained in the UCI Machine Learning Repository [70], that have been widely used to test Data Mining algorithms.

2. **Data Preprocessing and Normalization:** The data preprocessing methods prepare the data to be analysed. There are three main steps [111]: avoid misclassification, dimensionality reduce (through projections or feature selection techniques) and range normalization.

3. **Model Generation:** This is the most important part of the data analysis process. The model is created to find patterns within data. It is usual to use Machine Learning or other statistical techniques to generate the model [111].

4. **Model Validation:** Depending on the type of model, the validation process is different. This process gives the confidence of the model. It is usual to use validation with classifiers [111], however, for the clustering problem, validation is almost a blind process [203] (which is a consequence of the clustering nature).

5. **Model Application:** The goal of the model is to be applied in order to predict the behaviour of new inputs. There are several applications of Data Mining as will be explained later.

This work is focused on the Model Generation methods. These methods are based on Statistical Inference and Machine Learning techniques. Machine Learning approaches are based on a "machine" which receives a sequence of inputs, called data, and looks for patterns which can be used to predict or explain the behaviour of new inputs. Some applications of these models can be found in [203]: business, biology, music, human behaviour, games, amongst others. These techniques can be divided in four main categories [5]:

- *Supervised Learning:* A sequence of desired outputs is also given with the inputs. The goal of the machine is to learn how to produce the correct output given a new input. The output could be a class label (classification) or a real number (regression). Some examples of classical supervised methods are [111]: Decision Trees, Support Vector Machines and Neural Networks.

- *Reinforcement Learning:* The machine produces a set of actions which affects the effect of an environment. These effects generate a reward (or punishment) which must be maximized in the future through the machine decisions.

- *Game Theory Learning:* It is a generalization of reinforcement learning. In this case, the environment can contain other machines with the same characteristics.

- *Unsupervised Learning:* The machine simply receives the input. Its goal is to generate the labels from the input set. This work is focused on generating clustering algorithms for unsupervised learning through genetics algorithms. The main clustering techniques created in this work will be described in the following chapters.

As it was explained in the Introduction, the most common unsupervised data mining method is clustering. Another good example of unsupervised learning method is dimensionality reduction[5] which is the process of reducing the number of random variables under consideration in a dataset analysis. These techniques are also known as feature selection methods and are presented in the following section.

2.2 Data Extraction and Representation

Any Data Mining process stars with the data acquisition phase. Data have several representations and can be obtained from different sources. The Data Mining techniques strongly depend on the data information and structure. On the one hand, the kind of information provided by the data determines the technique which will be applied (e.g., clustering, classification, regression, among others). On the other hand, the structure defines the way to analyse the data (e.g.,

time series analysis, text mining, image segmentation, etc). Using this information, the analyser is able to determine the best technique to extract patterns from the data.

2.2.1 Types of data representation

Data structures can be divided into several categories. The most relevant which have been used in Data Mining are the following [111]:

- Static Data [70]: This kind of data is the most frequent. It is usually divided in numerical and categorical data and can be represented as a matrix. When data is categorical, it is usually converted to a numeric representation.
- Time Series [199]: Time Series data are similar to static data but it pays attention to the temporal component. These data are usually used to predict future behaviour of a set of events in one -or several, variables (e.g., stock market).
- Texts [26]: Text documents usually provide information which is analysed in a semantic context. Different techniques, such as Topic Detection and Tracking (TDT) or document summarization, are good examples about how these data need to be interpreted in a different way.
- Images [209]: Images have two main kind of analysis: detect different parts of an image (e.g. segmentation), or group images by similarity (e.g., discriminate images which have examples of cars, trains, etc. against other images). An image representation is usually a set of pixels.
- Stream data [9]: Stream data is a concept which describes those data sources which continually provide data. The algorithms used to analyse these kinds of data are usually focused on large data analysis or online analysis.

2.2.2 Databases

In addition to the data structure identification process, it is important to consider different databases where these data types can be found:

- UCI Machine Learning Repository [70]: the Center of Machine Learning and Intelligent Systems provides around 300 datasets used for Data Mining analysis. This repository is extremely useful in order to test new algorithms.
- R [166]: The R-project also provides different databases and packages which can generate synthetic data to test algorithms.
- Berkeley Segmentation Dataset [130]: This dataset is a benchmark for image segmentation analysis. It provides different images and a few human-based segmentations to evaluate new techniques.
- BioMed [6]: This database contains several references to medical and biomedical literature. These documents are usually papers or books about medical research.

2.3 Data Preprocessing and Normalization

Data Mining techniques need an intensive phase of data preprocessing. Initially the information must be analysed and stored in some kind of database system, cleaned and separated. This preprocessing phase is used to avoid outliers, misclassification and missing data. Methods such as histogram and statistical correlation are used to clean the dataset and reduce the number of variables [111]. Projections are also usual in dimension reduction, however, projection methods [55] such as PCA (Principal Component Analysis) or LDA (Lineal Discriminant Analysis) do not offer a complete perspective of the problem. These methods create new variables which are estimated from principal components or lineal projection trying to separate the data and reduce its dimension. Usually, these techniques lose the original information of the features which is unrecoverable once it is projected. It produces a reduction of the human interpretation of the Data Mining techniques applied and, sometimes, it is preferable to avoid them.

There are several techniques which reduce the feature sets to avoid projections. These methods apply a guided search among the different attributes looking for the most useful variables for the analysis. They are usually known as feature selection methods [105]. Some examples can be found in the work of Curiel et al. [49] where they apply genetic algorithms to simplify prognosis of endocarditis using a codification where each individual of the population is based on a set of features. Blum and Langley [30] show some examples of relevant features selections in different datasets and applied them to different machine learning techniques. They define different degrees of relevant features such as strong or weak relevant features. They also study some methodologies such as heuristic search, filters and wrapper approaches which are automatic feature selection methods usually validated by classification techniques. Some of these techniques usually introduce over-fitting to the model and are computationally expensive. Roth and Lange [174] apply these techniques to the clustering problem. Our previous work showed a new feature and simple selection methodology whose goal is to make an intensive reduction of the attributes dimension oriented to clustering analysis. It applies clustering methods to validate the chosen feature set [134].

Finally, the last step is related to normalization. It allows to compare data features with different kind of value range. Z-Score [36] and Min-Max [84] normalization methods are commonly used for preprocessing the data. Both normalization algorithms take the attribute records and they find a standard range for them. Min-Max has a fixed range, [0,1] (it is sensitive to outliers), while Z-Score depends on the mean and the standard deviation (it approximates the distribution to a normal distribution, it is usually used to avoid outliers). These algorithms obtain the normalized values from data using the following equations:

- **Min-max**: It computes maximum and minimum values of the attributes, and applies:

$$x' = \frac{x - min(X)}{max(X) - min(X)}$$

- **Z-Score**: It computes mean and standard deviation of the values, and applies:

$$x' = \frac{x - mean(X)}{SD(X)}$$

Due to data analysis depends of the data representation, it is important to consider the main modifications which can be found according to the source. The following section explains how these techniques are usually applied to text data, due to this dissertation considers this special case.

2.3.1 Text Preprocessing

Text-based documents have a special methodology for preprocessing. Once the set of documents is ready, they are grouped in a corpus of documents [26] which provides a conceptual representation of the database. There are several transformations which are usually applied to the corpus, in order to perform the analysis. The most commons are:

- Remove special characters, numbers, strip whitespaces, punctuations, etc.
- Remove stopwords (this step needs to specify the language).
- Transform strings to acronyms.
- Stemming the words, i.e., keep only the word root.

Once the data has been cleaned, a structure for the document is generated. The most frequent structure is the document-term matrix [26], where the corpus is represented as a matrix whose rows represent documents and whose columns represent possible terms to appear in the documents. The matrix values are the frequency of term appearance in the documents. This matrix is usually preprocessed removing sparse terms (i.e., infrequent terms). Also, the matrix representation is optimized using a Term Frequency-Inverse Document Frequency (TF-IDF) model. The former (TF) represents the number of occurrences of a word divided by the number of words of the entire document. The latter (IDF) is used to measure whether the term is common in the corpus. The equations associated with these two values are:

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) \ : \ w \in d\}}, \tag{2.1}$$

$$idf(t,C) = \log\left(\frac{|C|}{|\{d \in C \ : \ t \in d\}|}\right), \tag{2.2}$$

where t and w represent terms from a document (d), which belongs to the corpus (C). The function $f(t,d)$ is the frequency of a term in a document, and $|C|$ is the number of documents in the corpus. The final values for the matrix are calculated as follows:

$$tf - idf(t,d,C) = tf(t,d) \cdot idf(t,C). \tag{2.3}$$

This value is usually normalized in order to range it between $[0,1]$.

Other important representation for texts is the graph-concept representation [62]. This representation uses a hypernym tree for the corpus terms to generate semantic-relation levels. The hypernyms are usually extracted from a dictionary, such as Wordnet [63], or they can be generalized from categories of an encyclopaedia, such as DBpedia [15].

Once the data are ready for the analysis, the model generation phase starts. This work is focused on unsupervised learning techniques for model generation, specially clustering techniques, that are presented in the following sections.

2.4 Model Generation: Classical Clustering Techniques

Clustering has become an important field in Data Mining. It is used to find hidden information or patterns in an unlabelled dataset and has several applications related to biomedicine [205], marketing [82], image segmentation [159] and virtual worlds [25] amongst others.

Clustering techniques are frequently used in Data Mining and Machine Learning methods. A popular clustering technique is K-means. Given a fixed number of clusters, K-means tries to find a division of the dataset [126] based on a set of common features given by distances or metrics that are used to determine how the cluster should be defined. Other approximations, such as Expectation-Maximization (EM) [56], are applied when the number of clusters is unknown. EM is an iterative optimization method that estimates some unknown parameters computing probabilities of cluster membership based on one or more probability distributions; its goal is to maximize the overall probability or likelihood of the data being in the final clusters [152].

Since these techniques fix the number of clusters a priori, there are validation techniques such as cross-validation [104] which are used to improve the number of clusters selection (through metrics such as the Minimum Sum-of-Squares [147]).

Other research lines have tried to improve these algorithms. For example, some *online* methods have been developed to avoid the K-means convergence problem to local solutions which depend on the initial values [19]. These methods create the clusters adding a new data instance at each step and modifying the cluster structure with this new information. Some other improvements of K-means algorithm are related to deal the different kinds of data representation, for example, mixed numerical data [12] and categorical data [176]. There are also some studies comparing methods with different datasets, for example, Wang et al. [197] compare self-organizing maps, hierarchical clustering and competitive learning where establishing molecular data models of large size sets. Other approaches related to genetic algorithms, and directly related to this work, will be described in Section 2.6.

Machine learning techniques have also been improved through the k-means algorithm, for example, reinforcement learning algorithms [18, 75]; or using topological features of the data set [74, 75] which can also be helpful for data visualization.

The following sections introduces the two most classical clustering algorithms: Expectation-Maximization and K-means. The following section presents the Spectral Clustering algorithm which inspired this work.

2.4.1 K-means

K-means [125] is maybe the most popular and well known partitional clustering algorithm. It is a straightforward clustering guided method (usually by a heuristic or directly by a human) to group data in a predefined number of clusters. As it was described above, given a fixed number of clusters (k), K-means tries to find a division of the dataset [126] based on a set of common features given by distances (or metrics) that are used to determine what elements belong to each cluster. K-means has also been improved though different techniques, like genetic algorithms [22]. Algorithm 1 shows the pseudo-code for K-means algorithm [111].

K-means initially sets the centroids (line 1) and the elements are added to the cluster whose centroid is closer to them (lines 5 to 7). After, the new centroids position of the clusters are calculated (line 9) and again the closest elements are added (lines 5 to 7). It continues until the centroid position converge to a fixed point (line 2).

Algorithm 1 Pseudo-code for K-means algorithm

Input: A dataset of n elements $X = \{x_1, \ldots, x_n\}$ and a fix number of clusters k.
Output: A set of clusters $C = \{C_1, \ldots, C_k\}$ which partitionate X

1: Assign k records to be the initial cluster centroids. We define the set of centroids as $Y = \{y_1, \ldots, y_k\}$ and $Y' = \emptyset$
2: **while** $Y \neq Y'$ **do**
3: Set all $C_j = \emptyset$.
4: $Y' \leftarrow Y$.
5: **for all** $x_i \in X$ **do**
6: Calculate the minimal distance centroid to x_i. Let y_j be the minimal distance centroid to x_i.
7: Introduce x_i in C_j.
8: **end for**
9: Calculate the centroids of C and set $y_i \leftarrow centroid(C_i)$.
10: **end while**
11: **return** C

2.4.2 Expectation Maximization

Expectation Maximization (EM) [56] is used when the number of clusters is unknown. Initially, it takes the likelihood and tries to maximize it. The process consists on apply the two following steps iteratively until it converges:

- **Expectation step**: Fix a model (θ) and estimate missing labels ($\mathbf{y}$).
- **Maximization step**: Fix missing labels ($\mathbf{y}$)(or a distribution over missing labels) and find the model (θ) which maximizes the likelihood function ($L(\theta)$) of the data.

The likelihood function is defined by:

$$L(\theta) = p(\mathbf{X}, \mathbf{y}|\theta) = \prod_i p(\mathbf{x}_i, y_i|\theta)$$

Where θ is a model defining how each instance $\mathbf{x}_i$ is assigned to a label y_j. This algorithm begins with the definition of an initial model $\theta^{(0)}$ and constructs a sequence $\theta^{(0)}$,$\theta^{(1)}$,...,$\theta^{(t)}$,... of models with increasing value of likelihood. To simplify the calculation process, the logarithm of the likelihood function is used:

$$l(\theta) = \log L(\theta) = \sum_i \log p(\mathbf{x}_i, y_i|\theta)$$

EM also uses an auxiliary function, which depends on the model and the distribution of missing labels, defined by:

$$Q(\theta|\theta^{(m)}) = \sum_i Q_i(\theta|\theta^{(m)}) = \sum_i E_{y_i|\mathbf{x}_i,\theta^{(m)}}[\log p(\mathbf{x}_i, y_i|\theta)]$$

$$Q(\theta|\theta^{(m)}) = \sum_i \sum_{j=1}^{k} P(y_i = j|\mathbf{x}_i, \theta^{(m)}) \log p(\mathbf{x}_i, y_i|\theta) \tag{2.4}$$

This function is used to increment the likelihood of the estimator θ which is taken as a maximum of this function. Algorithm 2 shows the pseudo-code for EM algorithm.

Algorithm 2 Expectation Maximization Pseudo-code [81]

Input: A dataset of n elements $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. A convergence error value δ.
Output: A set of clusters $C = \{C_1, \ldots, C_k\}$ which partitionate X

1: Fix a number of cluster k {this value is estimated applying cross-validation and repeating this algorithm with different values of k}
2: Choose the initial model $\theta^{(0)}$.
3: Compute the initial log-likelihood $l^{(0)}(\theta^{(0)})$.
4: **repeat**
5: **E-step**: Calculate $\{\gamma_{ij}^{(m)}\}$ where $\gamma_{ij} = P(y_i = j|\mathbf{x}_i, \theta^{(m)})$. {For the m iteration}
6: **M-step**: Calculate $\theta^{m+1} = \text{argmax}_\theta \left(Q(\theta|\theta^{(m)})\right)$, where $Q(\theta|\theta^{(m)}) = \sum_i \sum_{j=1}^{k} \gamma_{ij}^{(m)} \log p(\mathbf{x}_i, y_i|\theta)$. {Extracted from equation 2.4}
7: **Convergence check**: Calculate $l^{(m+1)}(\theta^{(m+1)})$.
8: **until** $|l^{(m+1)} - l^{(m)}| < \delta$
9: Put $\mathbf{x}_i$ in C_j if $\gamma_{ij} = max(\{\gamma_{iq}\}_{q=1}^{k})$
10: **return** C

The m-th iteration of the E-step (line 5) produces a guess of the $n \times k$ membership-probabilities of the elements to the clusters $\{\gamma_{ij}\} = P(y_i = j|\mathbf{x}_i, \theta^{(m)})$, where γ_{ij} is the current guess of the probability that sample $\mathbf{x}_i$ came from the j-th cluster. The M-step (line 6) gives a closed-form solution to the new estimates of the estimator θ. It converges if the increment of the likelihood is lower than a value (δ) given.

2.4.3 Spectral Clustering

Spectral clustering methods are based on a straightforward interpretation of weighted undirected graphs as can be seen in [16, 150, 155, 193]. The Spectral Clustering approach is based on a Similarity Graph which can be formulated in three different ways (all of them equivalent [193]) of graphs:

1. **The ϵ-neighbourhood graph**: all the components whose pairwise distance is smaller than ϵ are connected.
2. **The k-nearest neighbour graphs**: the vertex v_i is connected with vertex v_j if v_j is among the k-nearest neighbours of v_i.

3. **The fully connected graph**: all points with positive similarity are connected with each other.

The main problem is how to compute the eigenvector and the eigenvalues of the Laplacian matrix of this Similarity Graph. For example, when large datasets are analysed, the Similarity Graph of the Spectral Clustering algorithm takes too much memory, it makes difficult the eigenvalues and eigenvectors computation. Some works are focused on this problem: von Luxburg et al. [193] present the problem, Ng et al.[155] apply an approximation to a specific case, and Nadler et al.[150] apply operators to get better results. The classical algorithms can be found in [193].

The theoretical analysis of the observed good behaviour of SC is justified using the perturbation theory [193, 150], random walks and graph cut [193]. The perturbation theory explains, through the eigengap, the behaviour of Spectral Clustering.

Spectral Clustering methods were introduced by Ng et al. in [155]. These methods apply the knowledge extracted from graph spectral theory to clustering techniques. These algorithms are divided in three main steps:

1. First, the algorithm constructs a graph using the data instances as nodes and applies a similarity measure to define the edges weights (see Algorithm 3, line 1). In this work a fully connected graph has been considered (the different types of graphs were previously explained). The measure used in this work is the Radial Basis Function (RBF) Kernel (which is the most usual approach taken in the literature) defined by:

$$s(x_i, x_j) = e^{-\sigma||x_i - x_j||^2} \tag{2.5}$$

   where σ is used to control the width of the neighbourhood.

2. Second, it studies the graph spectrum calculating the Laplacian Matrix associated to the graph (see Algorithm 3, lines 2 and 3). There are different definitions of the Laplacian Matrix. These definitions obtain different results when they are applied to the Spectral Clustering algorithm. They can be used to categorize the Spectral Clustering techniques as follows [193]:

   - **Unnormalized Spectral Clustering**: It defines the Laplacian matrix as:

$$L = D - W$$

   - **Normalized Spectral Clustering**: It defines the Laplacian matrix as:

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

   - **Normalized Spectral Clustering (related to Random Walks)**: It defines the Laplacian matrix as:

$$L_{rw} = D^{-1} L = I - D^{-1} W$$

   In these formulas I is the identity matrix, D represents the diagonal matrix whose (i, i)-element is the sum of the similarity matrix ith row and W represents the Similarity Graph (see Algorithm 3, line 2). Once the Laplacian is calculated (in Algorithm 3 the Normalized Spectral Clustering algorithm is used, however, in this case, to simplify, the eigenvalues

which are calculated are $1 - \lambda_i$ instead of λ_i, the eigenvectors do not change), its eigenvectors are extracted (see lines 4 and 5 of Algorithm 3). Some of the main problems of Spectral Clustering are related to the consistency of the two classical methods used in the analysis: normalized and un-normalized Spectral Clustering. A deep analysis about the theoretical effectiveness of normalized clustering over un-normalized can be found in [194].

3. And, finally, the eigenvectors of the Laplacian Matrix are considered as points and a clustering algorithm, such as K-means, is applied over them to define the clusters (see Algorithm 3, lines 7 and 8).

The Spectral Clustering algorithm which is used in this work is the Normalized Spectral Clustering Algorithm introduced by Ng [155] (see Algorithm 3).

Algorithm 3 Normalized Spectral Clustering according to Ng et al. (2001)[155]

Input: A dataset of n elements $X = \{x_1, \ldots, x_n\}$ and a fix number of clusters k.
Output: A set of clusters $C = \{C_1, \ldots, C_k\}$ which partitionate X

1: Form the affinity matrix $W \in R^{n \times n}$ defined by $W_{ij} = e^{-||x_i - x_j||^2/2\sigma^2}$ if $i \neq j$, and $W_{ii} = 0$.
2: Define D to be the diagonal matrix whose (i, i)-element is the sum of the i-th row of W.
3: Construct the matrix $L = D^{-1/2} W D^{-1/2}$.
4: Find $v_1, \ldots, v_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues) and form the matrix $V = [v_1 v_2 \ldots v_k] \in R^{n \times k}$ by stacking the eigenvectors in columns.
5: Form the matrix Y from V by renormalizing each row of V to have unit length (i.e. $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$).
6: Apply K-means (or any other algorithm) treating each row of Y as a point in R^k.
7: Assign the points x_i to cluster C_j if and only if the row i of the matrix Y was assigned to cluster j.
8: **return** C

Part of the present work is inspired by Spectral Clustering because we use a clustering technique which analyse a Similarity Graph. Nevertheless, in our case we are using different methods based on Genetic Algorithms to find the clusters, instead of the Laplacian matrix extracted from the Similarity Graph.

2.5 Model Generation: Graph-based and Complex Networks models

Graph theory is also an essential area of research in data analysis, especially in the last years with its application to manifold reconstruction [79] using data distance and graph representation to create a structure which can be considered as an Euclidean space (i.e., the manifold).

Graph models are useful for diverse types of data representation. They have become especially popular over the last years, being widely applied in the Social Networks area. Graph models can be naturally used in these domains, where each node or vertex can be used to represent an agent, and each edge is used to represent their interactions. Besides, algorithms, methods and graph theory have been used to analyse different aspects of the network, such as: structure, behaviour, stability or community evolution inside the graph [54, 68, 151, 200].

A complete roadmap to graph clustering can be found in [177] where different clustering methods are described and compared using different kinds of graphs: weighted, directed, undirected. These methods are: cutting, spectral analysis and degree connectivity (an exhaustive analysis of connectivity methods can also be found in [87]), amongst others. This roadmap also provides an overview of computational complexity from a theoretical and experimental point of view of the studied methods.

From previously described graph clustering techniques, a recent and really powerful ones are those based on Spectral Clustering which were previously introduced. In this section, several definitions of Graph Theory and an introduction of the Complex Network approach and the metric extracted from this field are presented.

2.5.1 Basic Definitions from Graph Theory

Defining and selecting an appropriate heuristic function is one of the most critical issues in any search algorithm. Our approach uses concepts and metrics extracted from graph theory. For this reason, and before describing it, some basic concepts are briefly introduced.

Definition 2.5.1 (Graph)**.** A graph $G = (V, E)$ is a set of vertices or nodes V denoted by $\{v_1, \ldots, v_n\}$ and a set of edges E where each edge is denoted by e_{ij} if there is a connection between the vertices v_i and v_j.

Graphs can be directed or undirected. If all edges satisfy the equality $\forall i, j,\ e_{ij} = e_{ji}$, the graph is defined as undirected.

The graph can also be represented through its adjacency matrix (the most usual approach) which can be defined as:

Definition 2.5.2 (Adjacency Matrix)**.** An adjacency matrix of G, A_G, is a square $n \times n$ matrix where each coefficient satisfies:

$$(a_{ij}) = \begin{cases} 1, & \text{if } e_{ij} \in E \\ 0, & \text{otherwise} \end{cases}$$

When it is necessary to work with weighted edges, a new kind of graph needs to be defined:

Definition 2.5.3 (Weighted Graph)**.** G is a weighted graph if there is a function $w : E \to R$ which assigns a real value to each edge.

Any algorithm that works with the vertices of a graph needs to analyse each node neighbours. The neighbourhood of a node is defined as follows:

Definition 2.5.4 (Neighbourhood)**.** If the edge $e_{ij} \in E$ and $e_{ji} \in E$ we say that v_j is a neighbour of v_i. The neighbourhood of v_i Γ_{v_i} is defined as $\Gamma_{v_i} = \{v_j \mid e_{ij} \in E \text{ and } e_{ji} \in E\}$. Then, the number of neighbours of a vertex v_i is $k_i = |\Gamma_{v_i}|$.

Also nodes can generate paths between them through their edges, a path is defined as follows:

Definition 2.5.5 (Path)**.** A Path of a graph between the nodes v_i and v_j is a set of edges which connects these two nodes. It will be denoted by P_{ij}.

And its length can be defined as:

Definition 2.5.6 (Path Length)**.** The Path Length is defined as the number of edges contained in the path. It will be denoted by $|P_{ij}|$.

It is also important to know the shortest path between two nodes, usually defined by:

Definition 2.5.7 (Shortest Path)**.** The Shortest Path is a minimum Path between two nodes. It should satisfy:

$$\min_{|P_{ij}|}\{P_{ij}|\ P_{ij} \in G\} \tag{2.6}$$

One is most important metrics of the graph is defined by its diameter:

Definition 2.5.8 (Graph Diameter)**.** The Graph Diameter is defined as the maximum shortest path of the graph.

Once the most general and simple concepts from graph theory are defined, we can proceed with the definition of some basic measures related to any node in a graph: the average path length, the clustering coefficient and the weighted clustering coefficient.

Definition 2.5.9 (Average Path Length)**.** Let G be a Graph and V its set of vertices. Let $d(v_i, v_j)$ be the shortest distance between v_i and v_j. The Average Path Length is defined by:

$$l_G = \frac{1}{n \cdot (n-1)} \cdot \sum_{i,j} d(v_i, v_j) \tag{2.7}$$

Definition 2.5.10 (Local CC [54])**.** Let $G = (V, E)$ be a graph where E is the set of edges and V the set of vertices and A its adjacency matrix with elements a_{ij}. Let Γ_{v_i} be the neighbourhood of the vertex v_i. If k_i is considered as the number of neighbours of a vertex, we can define the clustering coefficient ***(CC)*** of a vertex as follows:

$$C_i = \frac{1}{k_i(k_i-1)} \sum_{j,h} a_{jh}a_{ij}a_{ih}a_{ji}a_{hi} \tag{2.8}$$

The Local CC measure provides values ranging from 1 to 0. Where 0 means that the node and its neighbours do not have clustering features, so they do not share connections between them. Therefore, value 1 means that they are completely connected. This definition of CC can be extended to weighted graphs as follows:

Definition 2.5.11 (Local Weighted CC [20])**.** Following the same assumption of Local Clustering Coefficient definition, let W be the weight matrix with coefficients w_{ij} and A be the adjacency matrix with coefficients a_{ij}, if we define:

$$S_i = \sum_{j=1}^{|V|} a_{ij}a_{ji}w_{ij} \tag{2.9}$$

Then, the Local Weighted Clustering Coefficient can be defined as:

$$C_i^w = \frac{1}{S_i(k_i-1)} \sum_{j,h} \frac{(w_{ij}+w_{ih})}{2} a_{jh}a_{ij}a_{ih}a_{ji}a_{hi} \tag{2.10}$$

For this new definition, we are considering the connections between the neighbours of a particular node, but now we add information about the weights related to the original node. This new measure calculates the weights distribution of the node that we are analysing, and shows how good the connections of that cluster are. The following theorem proves that the weighted CC has the same value than the CC when all the weights are set to the same value:

Theorem 2.5.1. *Let G be a graph, A its adjacency matrix and W its weight matrix. If we set $w_{ij} = \omega \ \forall i,j$, them $C_i = C_i^w$.*

Proof. Following the definition of C_i^w we have:

$$C_i^w = \frac{1}{S_i(k_i-1)} \sum_{j,h} \frac{2\omega}{2} a_{jh}a_{ij}a_{ih}a_{ji}a_{hi}$$

Where $S_i = \sum_{j=1}^{|V|} a_{ij}a_{ji}\omega$. Replacing S_i, we have:

$$C_i^w = \frac{1}{\sum_{j=1}^{|V|} a_{ij}a_{ji}\omega(k_i-1)} \sum_{j,h} \omega a_{jh}a_{ij}a_{ih}a_{ji}a_{hi}$$

$$C_i^w = \frac{1}{\sum_{j=1}^{|V|} a_{ij}a_{ji}(k_i-1)} \sum_{j,h} a_{jh}a_{ij}a_{ih}a_{ji}a_{hi}$$

We also know that following the neighbour definition and the adjacency matrix definition: $k_i = \sum_{j=1}^{|V|} a_{ij}a_{ji} = |\Gamma_{v_i}| = |\{v_j \mid e_{ij} \in E \text{ and } e_{ji} \in E\}|$ And finally:

$$C_i^w = \frac{1}{k_i(k_i-1)} \sum_{j,h} a_{jh}a_{ij}a_{ih}a_{ji}a_{hi}$$

Which proves theorem 1.

As a corollary to this theorem, if $CC_i^w = 1 \Rightarrow CC_i = 1$.

Finally, if we want to study a general graph, we should study its Global CC:

Definition 2.5.12 (Global CC [54, 20])**.** The clustering coefficient of a graph can be defined as:

$$C = \frac{1}{|V|} \sum_{i=0}^{|V|} C_i \tag{2.11}$$

Where $|V|$ is the number of vertices.

The Global Weighted Clustering Coefficient is:

$$C^w = \frac{1}{|V|} \sum_{i=0}^{|V|} C_i^w \tag{2.12}$$

The main difference between Local CC, Local Weighted CC and Global CC is that, the first one can be used to measure the local connectivity of a node in the graph, the second one is used to calculate the density of these connections using the edge weights, and the last one provides global information about the connectivity in a graph. In real complex problems only the two initial measures can be used, whereas the third one is usually estimated [191].

2.5.2 Complex Networks Analysis

In network analysis, is common to use a graph representation, especially for the Social Network approaches where users are connected by affinities or behaviours. This approximation has been studied in some of the small world networks based on two main variables: the average distance between elements, and the clustering coefficient of the graph [54, 151, 200].

The present work is closer to the network approach because our algorithm looks for subgraphs in a graph whose elements share similar features. In an initial study of the problem [22], an evolutionary approach was adopted based on the K-means algorithm applied to community finding approach (which is also a clustering problem applied to a graph representation).

Other similar approximations related to the finding-community problem can be found in Reichardt and Bornholdt[169] where different statistical mechanics for community detection are used. Pons and Latapy[165] use random walks to compute the communities. However, we decided to use genetic algorithms because we are interested in optimization methods for tuning up the definition of our clusters, allowing to adapt the size and membership of these clusters using metrics and features selected from graph characteristics.

Finally, another work based on metrics used to measure the quality of the communities was developed by Newman and Girvan [154], and Clauset et al.[41] designed different metrics that can be used to find the structure of a community in very large networks. Genetic algorithms have also been applied to find communities (or clusters) through agglomerative genetic algorithms [119] and multi-objective evolutionary algorithms [103] amongst others.

2.6 Model Generation: Genetic Algorithms for Clustering

Over the last years evolutionary clustering has attracted much research interest, yielding a large literature corpus. Evolutionary Computation is a vast field that includes many families of algorithms, all of them inspired in natural selection. Perhaps the most popular EA for clustering is Genetic Algorithms (GAs), where a population of candidate solutions is codified in strings named chromosomes. Then GAs apply a set of genetic operators (typically mutation and crossover) and a stochastic selection operator based on a fitness function to breed the next algorithm iteration. Hruschka et al. [90] presents a complete survey on this topic.

Genetic algorithms have been traditionally used in optimization problems, as was mentioned before. These algorithms have also been used for general data and information extraction [71]. The operators of the genetic algorithms can also be modified. Some examples of these modifications can be found in [164] where Poli and Langdon improved the algorithm using backward-chaining, creating and evaluating individuals recursively reducing the computation time. Other

applications of genetic clustering algorithms can be found in swarm systems [110], software systems [58], file clustering [65] and task optimization [163], amongst others.

The genetic clustering approximation tries to improve the results of the clustering algorithm using different fitness functions to tune up the cluster sets selection. In [44], Cole developeddifferent approaches of the genetic clustering problem, especially focused on codification and clustering operations. There is also a deep revision in [90] where Hruschka et al. provide a complete up to date state of the art in evolutionary algorithms for clustering.

There are several methods using evolutionary approaches from different perspectives, for example: Aguilar [11] modifies the fitness considering cluster asymmetry, coverage and specific information of the studied case; Tseng and Yang [190] use a compact spherical cluster structure and a heuristic strategy to find the optimal number of clusters; Maulik and Bandyopadhyay [132] use the clustering algorithm for metric optimization trying to improve the cluster centre positions; Shi et al. [180] based the search of the genetic clustering algorithm in their Extend Classifier Systems which is a kind of Learning Classifier System, in which a fitness of the classifier is determined by the measure of its prediction's accuracy; Das and Abraham [50] use Differential Evolution, a method that optimizes a problem by iteratively trying to improve a candidate solution with regards to a given measure of quality.

Some of those previous methods are based on K-means, for example: Krishna and Murty [107] replace the crossover of the algorithm using K-means as a search operator, and Wojciech and Kwedlo [202] also use differential evolution combined with K-means, where it is used to tune up the individuals obtained from mutation and crossover operators. Finally, other general results of genetic algorithm approaches to clustering can be found in [7]. There are also other complete studies for multi-objective clustering developed by Handl et al. [85] and for Nearest Neighbour Networks developed by Huttenhower et al. [94].

2.6.1 MOGA for Clustering

This work applies a Multi-Objective Genetic Algorithm (MOGA) [53] approach. This approach is characterized by the capability to use opposite objectives in the same fitness function. The evolution of the individuals defines a Pareto Front where the best fitness values according to the metrics are found. These solutions are called **dominant solutions** and define a set of possible solutions to the problem.

MOGAs have been applied to several clustering problems [90]. There are usually two main approximations: some works generate a new MOGA to create a new clustering algorithm [149], while others apply classical MOGAs to solve the problem of minimizing some cost functions which are the objectives of the fitness function [113]. The most classical MOGAs are SPEA2 (Second version of the Strength Pareto Evolutionary Algorithm [212]), NSGA-II (Nondominated Sorting Genetic Algorithm [52]) and PESA [46] amongst others. These algorithms have been applied in clustering problems with different results [113]. NSGA-II and SPEA2 have demonstrated to achieve good results applied to clustering problems, however, SPEA2 usually defines a better Pareto Front than NSGA-II [212]. This work is based on a MOGA implementation which optimizes two objectives using SPEA2.

Another approach to evolutionary clustering with GAs comes from Multi-Objective Genetic Algorithm (MOGA). In this approach, the selection of the individual does not depend on one

criteria, but several ones. Most of the approaches to multi-objective evolutionary clustering use, with different names, inter and intra cluster distances, i.e., they try to minimize the distance between data and their cluster centroids, while minimizing the distance among the clusters centroids. Some authors claim the superiority of this approach, for instance, Ripon and Kwong [172] stated that traditional single objective algorithms suffer premature convergence that multi-objective algorithms solve. It is clear that sometimes using a single criteria loses important pieces of information that would be exploited in benefit of the search.

There are some proposals of MOGAs with adaptive number of clusters. Handl et al. proposed the Multi-Objective Clustering with automatic K-determination (MOCK) [86], a graph-oriented clustering algorithm on a MOGA. In this approach the chromosomes represent non-weighted graphs with an integer representation. Each loci represents a data instance and the allele a link to another instance. With this representation, a chromosome may contain several subgraphs, i.e., graphs without links to other graphs. These isolated subgraphs represent the clusters. Despite the graph-based representation, this approximation cannot be considered spectral clustering because of the lack of spectral analysis. Matake et al. proposed an improvement of MOCK [131] to compute k more efficiently and make the algorithm well suited for large datasets.

Another example of adaptive k multi-objective clustering algorithm is the Variable-Length Real Jumping Genes Genetic Algorithm (VRJGGA), proposed by Ripon et al. [172]. VRJGGA is an adaptive version of another algorithm named JGGA. It uses a Variable-Length Genetic Algorithm with a standard cluster centroid representation in a chromosome of floats. The variance of chromosome lengths is introduced with two custom genetic operators: cut-and-paste and copy-and-paste.

On the contrary than previous partitional clustering algorithms, Banerjee [17] used a MOGA to solve the fuzzy clustering problem with adaptive k and noisy data. This approach uses a quite complex representation scheme with each individual divided into two independent strings: one distinguish between clean and noisy data while the other one keeps the result of the partition.

Multi-objective spectral clustering is a recent topic with a scarce literature. One example is Wang [198], who proposed an evolutionary multi-objective spectral algorithm clustering algorithm for datasets that contain different views of the same data, for instance, because data come from heterogeneous sources. As a consequence, the dataset is represented by means of several graphs. In this context the algorithm is able to automatically determine k by means of Pareto optimization.

To the author's knowledge, the only attempt to address spectral clustering with multi-objective computational intelligence used Harmony Search Algorithm (HSA), this is a search method inspired by musicians improvisation that has an increasing number of applications. Li et al. proposed the Spectral Clustering-based Adaptive Hybrid Multi-Objective Harmony Search Algorithm (SCAH-MOHSA) [114], which is a complex algorithm for community detection in graphs; it uses spectral clustering with a Multi-Objective HSA and local search.

2.6.2 Other Bio-Inspired Approaches: ACO

Ant Colony Optimization (ACO) has become a promising field for data mining problems. In this context, ACO algorithms combine the ants foraging behaviour to generate patterns that describe

the data according to a supervised or unsupervised learning criteria -depending on the type of algorithm, classification or clustering, respectively. ACO algorithms have produced promising results. Kao and Cheng [100] introduced a centroid-based ACO clustering algorithm; and Ashok and Messinger focused their work on graph-based clustering [14]; several other approaches are discussed in [95].

There are also approaches that combine ACO with classical classification algorithms in order to improve their results. Some of these techniques, for example, optimize the parameter selection for the classifier (e.g., for SVMs [210]), other are focused on the feature selection process for the data preprocessing phase [57] and others are hybrid approaches [88].

In this work we have used ACO algorithms to test new bio-inspired approaches and to extent the clustering environment to other graph-based bio-inspired approaches.

2.7 Model Generation: Data Stream and Online Clustering Techniques

One of the current main challenges in Data Mining is to analyse massive data online. Due to classification requires a previous labelling process, these methods need high efforts for real-time analysis. However, due to unsupervised techniques do not need this previous process, clustering methods are a promising field for real-time analysis.

When Data Streams are analysed, it is important to consider the analysis goal, in order to determine the best type of algorithm to be used. We could divide data stream analysis in two main categories:

- Offline analysis: we consider a portion of data (usually large data) and apply an offline clustering algorithm to analyse this data.
- Online analysis: the data are analysed in real-time. These kinds of algorithms are constantly receiving new data instances and are not usually able to keep past information.

2.7.1 Offline Clustering Analysis

Large data analysis, as was mentioned above, is a current challenge for clustering and other Data Mining methods. The new databases contain lots of data instances which need to be analysed in a competitive time.

The classical clustering processes used to analyse datasets have been adapted in order to improve their scalability and parallelism during the whole processes. This philosophy gain ascendancy with the MapReduce [51] algorithm and the Hadoop [185] framework which implements this system. Current tools are based on Spark [207] which improves several features of MapReduce.

MapReduce is a model used to process and analyse large datasets using a parallel and distributed algorithm over a computer cluster. It is divided in two main steps:

- The "Map" step: The input is divided in smaller subproblems. These subproblems are concurrently processed by each cluster node. These subproblems may also be divided generating a tree structure. Once a node has complete the process, it sends the answer to its master node.
- The "Reduce" step: The master node combines all the answers from the subproblems to generate the final solution.

There are several adaptations of clustering algorithms to the MapReduce model, the most relevant ones are related to K-means [211], EM [39] and Spectral Clustering [38] algorithms.

2.7.1.1 K-means with MapReduce

K-means has been optimized through the MapReduce philosophy [211]. The algorithm scales in two main steps of K-means:

- Associate the data to the closest centroids.
- Calculate the cluster centroid.

The MapReduce version of the algorithm is divided in three main steps:

1. Initialization: The dataset is divided in blocks and the initial centroids are set.
2. Data Association (Map): the data of each block is associated to the closest centroid. In this case, each node associates the data of one block and all nodes share the set of centroids.
3. Centroid update (Reduce): Each node receives all the data which have been assigned to a centroid and update the centroid position.

After step 3), the centroids set is updated and steps 2) and 3) are repeated until the algorithm converges or a maximum number of iterations is achieved (see Figure 2.1 for an example).

2.7.1.2 EM with MapReduce

Expectation-Maximization is clearly divided in two steps. This division allows to use the same idea used with K-means using the two steps of EM [39]. In this case, the process is:

1. Initialization: The data are divided in blocks, too.
2. Expectation step (Map): Fixing the estimator (θ), the missing labels ($\boldsymbol{y}$) are estimated per block. In this case all blocks share a estimator.
3. Maximization step (Map and Reduce): Fixing the labels, the model will update the values according to the likelihood ($L(\theta)$) function. Part of this step might be calculated during the map process (e.g., partial operations which depend on the estimator). The updating process will be realized on the reduce process, producing new parameters for the E-step.

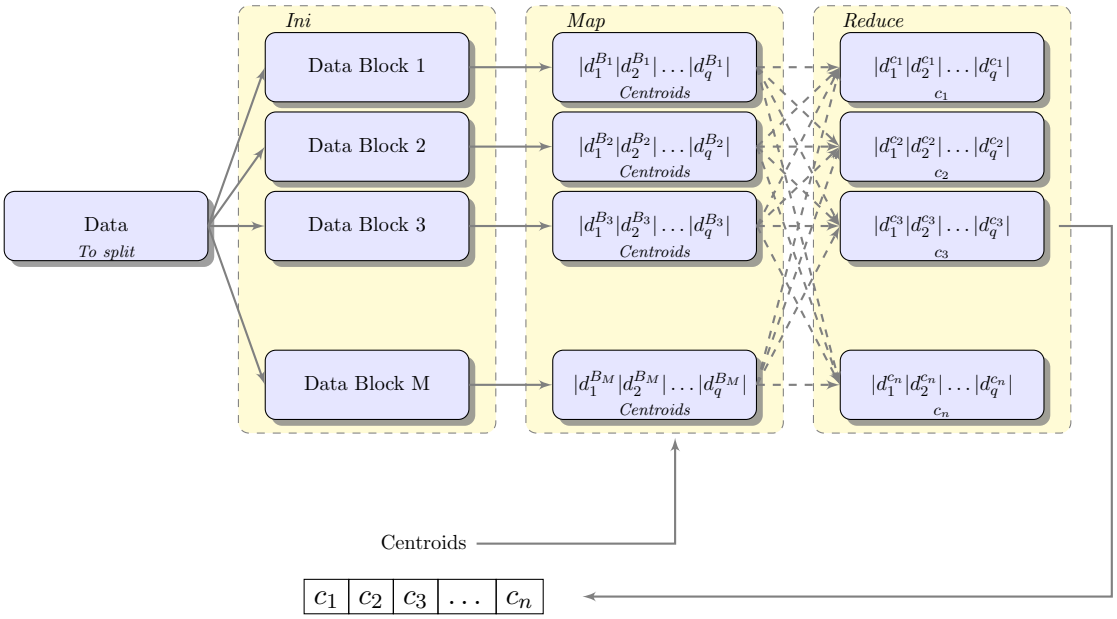


Figure 2.1: MapReduce Scheme for K-means.

A good example for the application is the Gaussian Mixture Model (GMM). In this case, we have the parameters μ (mean) and $\mathbf{\Sigma}$ (standard deviation) for the Gaussian sets and we need to calculate the values of γ_{ij} (partial values of probability) for the M-step. Let $p^{(j)}(y) = \sum_{partial} \gamma_{ij}$, then the mapper will associate the data and calculate the partial sums for μ and $\mathbf{\Sigma}$ in the following way:

$$\mu_{partial}^{(j)} = \sum_{partial} (\gamma_{ij} \cdot x_i) \tag{2.13}$$

$$\mathbf{\Sigma}_{partial}^{(j)} = \sum_{partial} (\gamma_{ij} \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^t), \tag{2.14}$$

While the reducer will update the values as follows:

$$\mu_i = \frac{\sum_{blocks} \mu_{partial}^{(j)}}{\sum_{blocks} p^{(j)}(y)} \tag{2.15}$$

$$\mathbf{\Sigma}_i = \frac{\sum_{blocks} \mathbf{\Sigma}_{partial}^{(j)}}{\sum_{blocks} p^{(j)}(y)} \tag{2.16}$$

2.7.1.3 Spectral Clustering with MapReduce

The Spectral Clustering algorithm has been successfully extended to the MapReduce paradigm [38]. The main motivation of this extension is the memory usage reduction (related to the similarity graph and the Laplacian matrix) and the eigenvectors computation. The MapReduce process is usually replicated during different steps of the algorithm.

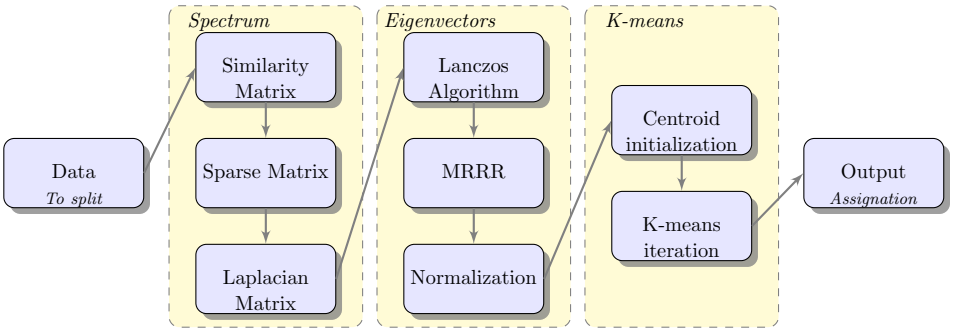


Figure 2.2: MapReduce process for Spectral Clustering.

The first step is the generation of the Similarity Graph and the Laplacian calculation. This step is parallelized including the generation of a sparse matrix which is helpful during the eigenvectores processing phase. The algorithm can be divided as follows (see Fig. 2.2):

- Similarity Graph (Map): the Similarity Graph is calculated comparing the data instances, this process is parallelized by sections.
- Construct the sparse matrix (Reduce): the t-closest instances are chosen for each node creating a sparse matrix.
- Laplacian Matrix Generation (Reduce): the Laplacian matrix is calculated using multiplications in order to keep the matrix in the distributed file system.

Once this first phase has been completed, it is needed to calculate the eigenvectors of the matrix. With this purpose, the algorithm used to identify the eigenvectors is an iterative algorithm called Lanczos Algorithm [72] (see Algorithm 4). This algorithm takes the Laplacian matrix and iteratively generates a triangular matrix called T (associated with a vector matrix called Q) which satisfies that each pair $< eigenvalue, eigenvector >$ of L (i.e., $(\lambda_k, Q_{n\times m}\vec{v}_k)$) corresponds to the pairs $< eigenvalue, eigenvector >$ $(\lambda_k, \vec{v}_k)$ of T. This pair can be extracted using a QR iteration. In this case the algorithm called Multiple Relatively Robust Representation [28] (MR^3) is applied (see Algorithm 5). This algorithm extracts the eigenvectors of T.

The Lanczos algorithm uses an iterative method (lines 3 to 7) to generate a tridiagonal matrix (line 9) whose eigenvector are related to the original matrix eigenvectors. The parallelization is performed in the matrix-vector product (line 3). The rest of the operations are computed locally.

The MR^3 algorithm divides the tridiagonal matrix generated by the Lanczos algorithm (line 1). Then it generates a factorization to calculate the first eigenvector (lines 3-5). With this information it generates a queue of factorization with the index of the eigenvalues (line 5). After, it recursively calculates the value of the eigenvectors using the factorizations generated. If the eigenvector can be calculated (lines 10 and 11) it is calculated, else, it add a new element to the queue (lines 13-16). This generates a tree of values which is represented in Figure 2.3.

Finally, once the k-first eigenvectors have been extracted, K-means (with MapReduce) is apply to the projection.

Algorithm 4 Pseudo-code for Lanczos algorithm

Input: L Laplacian matrix.
Output: T tridiagonal matrix; Q matrix

1: $\vec{q}_0 = 0$; $\beta_0 = 0$; $\vec{q}_1 =$ random vector;
2: **for** $k = 1$ **to** M **do**
3: $\quad \vec{w}_k = L_{\vec{q}_k} - \beta_k \vec{q}_{k-1}$
4: $\quad \alpha_k = \langle \vec{w}_k, \vec{q}_k \rangle$
5: $\quad \vec{w}_k = \vec{w}_k - \alpha_k \vec{q}_k$
6: $\quad \beta_{k+1} = ||\vec{w}_k||_2$
7: $\quad \vec{q}_{k+1} = \vec{w}_k / \beta_{k+1}$
8: **end for**
9: **return** $Q_{n \times m} = (\vec{q}_1, \ldots, \vec{q}_m)$, $T = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \ldots & 0 \\ 0 & \beta_3 & \ldots & \ldots & 0 \\ \vdots & \ddots & \vdots & \beta_{m-1} & 0 \\ \vdots & \ldots & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & \ldots & 0 & \beta_m & \alpha_m \end{pmatrix}$.

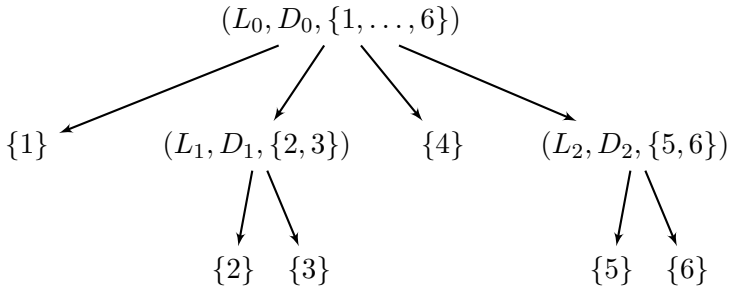


Figure 2.3: Tree representation of MRRR recursive algorithm.

Algorithm 5 Pseudo-code for MRRR algorithm

Input: T symmetric tridiagonal (irreducible). Γ_0 index of desired eigenpairs. *tol*: error tolerance (default 10^{-3}.

Output: $(\lambda_j, \vec{v}_j)$, $j \in \Gamma_0$ computed eigenvectors.

Split T into irreducible subblocks $T_1, \ldots, T_l$:
for all T_i, $i = 1, \ldots, l$ **do**
    Choose μ_i and calculate L_0 and D_0 such that

$$L_0 D_0 L_0^t = T_i + \mu_i I,$$

    is a factorization that determines the eigenpairs λ_j and $\vec{v}_j$, $j \in \Gamma_0$ to high accuracy
    Compute the eigenvalues of $L_0 D_0 L_0^t$
    Create a queue Q initialized as $Q = \{(L_0, D_0, \Gamma_0)\}$
end for
Recursively, while queue Q is not empty:
Remove an element (L, D, Γ): Partition the computed eigenvalues in nodes $\Gamma_1, \ldots, \Gamma_h$ according to the tolerance *tol*.
for all Γ_c, $c = 1, \ldots, h$ **do**
    if $|\Gamma_c| = 1$ with eigenvalue λ_c **then**
        Compute the eigenvector $\vec{v}_c$
    else
        Pick τ_c near the node and compute

$$LDL^t - \tau_c I = L_c D_c L_c^t$$

        "Refine" the eigenvalues $\lambda_c - \tau_c$ in order to improve its acuracy with respect to $L_c D_c L_c^t$.
        Set $\lambda_c = \lambda_c - \tau_c$ refined.
        Add (L_c, D_c, Γ_c) to Q.
    end if
end for

2.7.2 Online Clustering Analysis

The other important challenge of large data analysis is the online analysis. Clustering methods have become promising techniques in this field, due to these algorithms can deal with unlabelled data. Usually the online analysis is focused on data streams.

Data streams generate data continually. The idea behind the online clustering algorithm is to analyse this data using real-time techniques. These techniques usually need to deal with large data quantities which produces several limitations on the algorithm representation. The most relevant limitations of these systems are:

- The data order matters and can not be modified.
- The data can not be stored or re-analysed during the process.
- The results of the analysis depends of the time the algorithm have been stopped.

These limitations also opens new research ideas, such as, trend identification in the clustering behaviour. Due to the clustering results are constantly modified, the trend of the new results produce an interesting post analysis phase which allows the analyst to predict where the clustering algorithm is going.

The main problem of these algorithms is that they need a specific space to update the information. This limits the possibilities of the new algorithm, producing for example, that medoid-based clustering algorithms could not be adapted to this kind of analysis [80].

One of the main tools used for online clustering analysis is the Massive On-line Analysis (MOA) tool. This framework provides the following online clustering algorithms:

- ClusTree [106]: This online algorithm iteratively update the information of the clusters. It is able to consider the speed of the data stream generating the concept of the age of the object. It also maintains stream summaries.
- CluStream [10]: This algorithm combine offline clustering and online clustering in order to provide partial clustering solutions which measure the evolution of the clusters.
- Den-Stream [35]: This algorithm tries to identify shapes during the clustering process using a micro-clustering level. This algorithm keeps information of the micro-clusters composing the shape form of the shape-clusters.

All of these algorithms have been designed in order to identify more properties of the data stream than the final clusters in a specific moment. In this work, we have specially focused on the classical online clustering implementation which is describe in the next section.

2.7.3 Classical Online Clustering

The classical online clustering implementation is a based on a K-means algorithm [19]. It tries to reduce the cost function of K-means using a Gradient Descent approach.

The algorithm takes the following performance function as a starting point:

$$J = \sum_{i=1}^{N} \min_{j=1}^{k} ||x_i - c_j||^2 \tag{2.17}$$

Where x_i is a data instance, c_j is a centroid and the operation $||\cdot||$ is the norm. The equation represents that the algorithm tries to minimize this cost function. In order to perform this minimization, it calculates how the centroid should be minimized according to the Gradient Descent methodology.

With this target it calculates the closest centroid for each data instance as follows:

$$q^* = \arg\min_{q=1}^{k}(||x_i - c_q||) \tag{2.18}$$

Then, the algorithm updates each centroid using the gradient of the cost function, which is calculated as:

$$\nabla J = \frac{\partial J}{\partial c_{q^*}} = (x_i - c_{q^*}) \tag{2.19}$$

Using this information, the centroids are updated as follows:

$$c_{q^*}^{(new)} = c_{q^*} - \zeta(x_i - c_{q^*}) \tag{2.20}$$

Where ζ is the learning rate and is usually set to 0.05 or $1/N_i$, where N_i is the number of instances until that instance appears.

There are some variations of this algorithm. The most relevant are presented by Barbakh and Fyfe [19] which introduce three algorithms based on these ideas but using different cost functions. These algorithms are more robust according to the solution than the classical one.

This work explores this methodology in order to create a continuity-based clustering algorithm using online clustering.

2.8 Applications

Once the models are generated and validated, they are usually applied to a specific field. These fields are really varied. During this thesis, we have developed several works that can be seen as applications of Data Mining techniques (specially clustering algorithms). The following fields are the most relevant:

- **Robosoccer**: In [97], classification and clustering techniques are used to identify team behaviours and discriminate them according to this criterion. This work also studies the most successful behaviour during different robosoccer leagues.
- **Soccer**: [135] analyses the FIFA World Cup 2010. This work begins with a previous features selection analysis [134] which helps to determine the most relevant features for the clustering process. The clusters are used to identify different behaviours and are applied in order to determine how the Spanish strategy was successfully used during the 2010 World Cup.

- **Eurovision**: In [24] a genetic k-adaptive clustering algorithm for community detection is designed in order to identify different communities which vote in a similar way. This methodology tries to discover how the different countries of Eurovision Contest Song formed alliances. This work was started in [25] where the clustering algorithm was not k-adaptive.
- **Baseball**: In [145] the idea is to predict baseball results. The model generated uses time-series clustering in order to include pass information of teams and matches.
- **Twitter**: In [141] the main goal is to discriminate meta-topics of different tweets, in order to group them. In this case, we use DBpedia and LSA to determine the tweet category and categorized those concepts which do not belong to DBpedia.
- **Image Segmentation**: [136] presents the application of a new genetic clustering algorithm to image segmentation. The results show that the algorithm performs the results of classical clustering algorithms.
- **Marketing**: [183] applies classification to analyse sentiment in Twitter. This analysis is used to identify the best classifier in this field. The analysis is extended in [184]. This last work also include a Social Network Analysis based on communities to measure propagation. Also a parallel work is applied combining clustering [156] which is extended in [23].
- **Videogames**: [146] presents a new game called Dream which is a RPG-action game. Dream has been designed to extract data of different gameplays in order to analyse this data. The analysis is focused on the player evolution. It discriminate different player profiles according to their learning abilities. The model used to discriminate these profiles is based on time series clustering.

Also one of the most important applications of Genetic Graph-based Clustering is related to Biomedical summarization. The next section presents all the areas which have been relevant in order to study this problem.

2.9 Applications in Biomedical Text Summarization

Our work takes ideas and techniques from different fields of Artificial Intelligence, including Text Summarization, Graph Clustering and Evolutionary Algorithms. In the next subsections, we review the related work in such fields that is close to our work. We also present some related works that have applied Evolutionary Algorithms to NLP tasks.

2.9.1 Biomedical Text Summarization

Text summarization may be defined as the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks) [128]. There are two main approaches to the task of automatic summarization: extraction and abstraction. Extractive methods construct the summaries by selecting the most relevant sentences in the original documents, while abstractive ones build an internal representation and use natural language generation (NLG) techniques to write the summaries, so that

abstracts may contain novel sentences, unseen in the original sources. Abstractive approaches require complex semantic representation, inference and natural language generation, which have not still reached a mature stage nowadays [204]. For this reason, most works in automatic summarization focus on extractive methods.

Traditional summarization systems include computing some simple heuristic rules to estimate the relevance of sentences (such as the position of the sentence in the document or the presence of some cue words [60, 32]), counting the frequency of the words in the document to identify central terms [123], or training different machine learning models to deal with summarization as a classification task [108]. Recently, graph-based methods have attracted the interest of the summarization research community. Graphs allow for a more complete representation of text than traditional vectorial models that reflects the interaction between the different textual and semantic units (i.e., words or sentences). Graph-based methods usually represent the documents as graphs, where the nodes correspond to text units (such as words, phrases, sentences or even paragraphs), and the edges represent cohesion relationships between these units, or even similarity measures between them (e.g. the Euclidean distance). Once the graph that represents a document is created, the salient nodes are located in the graph and used to extract the corresponding units for the summary. Two commonly used metrics to identify salient information in this graph-based representation are degree centrality and eigenvector centrality [31], both based on connectedness.

LexRank [62] is a well-know example of a centroid-based method to multi-document summarization. It creates an undirected graph, where the nodes are the sentences (represented by their TF-IDF vectors) and the edges represent the cosine similarity between them. A very similar method is proposed by Mihalcea and Tarau [148] to perform mono-document summarization. As in LexRank, the nodes represent sentences and the edges represent the similarity between them, measured as a function of their content overlap. Litvak and Last [120] proposed an approach that uses a graph-based syntactic representation for keyword extraction, which can be used as a first step in summarization.

Summarization in the biomedical domain usually adapts generic approaches to work with domain-specific knowledge. In this line, [168] adapts the lexical chaining approach [21] to work with concepts from the Unified Medical Language System (UMLS) [4].

BioSquash [181] is a question-oriented extractive system for biomedical multi-document summarization. It constructs a semantic graph that contains concepts of three types: ontological concepts (general ones from WordNet and specific ones from the UMLS), named entities and noun phrases.

In [206], Yoo et al. represent a corpus of documents as a graph, where the nodes are the Medical Subject Headings (MeSH) [1] descriptors found in the corpus, and the edges represent hypernymy and co-occurrence relations between them. They cluster the MeSH concepts in the corpus to identify sets of documents dealing with the same topic and then generate a summary from each document cluster.

Fiszman et al. [66] propose an algorithm that makes use of semantic predications provided by SemRep [171] to interpret biomedical text and on the use of lexical and semantic information from the UMLS to produce abstracts from biomedical scientific articles. This same method is adapted in a later work to summarize drug information in MEDLINE citations [67].

Ling et al. [118] focus on the genomic domain, and present a system that ranks sentences according to three features: the relevance of six gene aspects, such as the DNA sequence, the relevance of the documents where the sentences are taken from, and the position of the sentences in the document.

More recent is the work of Shang et al. [178], where the aim is to combine information retrieval techniques with information extraction methods to generate text summaries of sets of documents describing a certain topic. To do this, they use SemRep to extract relations among UMLS Metathesaurus concepts and a relation-level retrieval method to select the relations more relevant to query concepts. Finally, they extract the most relevant sentences for each topic based on the previous ranking of relations and the location of the sentences in different sections of the document.

2.9.2 Evolutionary Algorithms in Natural Language Processing

Evolutionary algorithms have been successfully applied to different NLP problems, from grammar induction to machine translation, through parameter optimization and search [33].

Smith and Witten [186], for instance, describe a genetic algorithm for grammar induction. The genotype is a context-free grammar whose fitness is evaluated on the basis of how well it covers a training set of sample strings. Selection is performed in inverse proportion to the grammar's size, while mutation is implemented by randomly choosing one grammar (individual).

Litvak et al. [121] propose a language-independent approach for extractive summarization based on the linear optimization of several sentence ranking measures using a genetic algorithm. An individual here is a vector of the weights of the different sentence ranking measures; and selection retains the best fifth of the individual solutions (i.e., those getting the maximal ROUGE value).

Rodríguez et al. [173] evaluate different implementations of evolutionary algorithms to find the alignment between two sentences for being used in statistical machine translation. Hall and Klein [83] propose a generative phylogenetic model for automatically identifying cognate words from unaligned word lists, given only the known family tree of languages.

In [13], genetic algorithms are also applied to two fundamental NLP applications: tagging, i.e., assignment of lexical categories to words; and parsing, i.e., determination of the syntactic structure of sentences.

2.9.3 Learning Ranking

In order to decide the number of clusters for the summarization process, this work has also been focused on learning to rank, which is common in Document Retrieval.

Document Retrieval process requires to rank the documents by relevance. The ranking is decisive because it determines the most relevant documents for the users. Several ranking models have been recently developed with this goal. Using the amount of available data, this model follows a traditional machine learning methodology, in order to generate an effective ranking

solution. This machine learning techniques are called "learning-to-rank" methods [122] and they are usually based on classification.

Although there are several methods to deal with the ranking problem, this work is focused on pairwise approach [122]. This approach does not work with the prediction of the relevance degree of each document, instead, it considers the relative order between two documents. In this sense, pairwise is closer to the concept of "ranking".

There exists several pairwise ranking algorithms, the most classical algorithms are:

- **Ordering with Preference Function** [43]: this approach uses a loss function which is trained to estimate the best of two elements in ranking.
- **SortNet (Neural Network-Based Sorting Algorithm)** [170]: This algorithm also learns a preference function, however, this function is represented by a Neural Network.
- **RankNet (Learning to Rank with Gradient Descent)** [34]: In this approach, a scoring function is included and the model is generated using information about the difference between the scores of each pair of documents.
- **FRank (Ranking with a Fidelity Loss)** [189]: This methodology includes a fidelity loss function to avoid the problems related to the boundary of the loss function of RankNet.
- **SVM-Rank**[98]: It applies the SVM technology to perform pairwise classification. This methodology is used in this work to help the clustering algorithm to choose the best number of clusters.

Chapter 3

The Genetic Graph-based Clustering Approaches

"All he had to do was watch the game and understand how things worked, and then he could use the system, and even excel."

- Orson Scott Card

This chapter introduces the Genetic Graph-based Clustering (GGC) algorithm, that is mainly based on Genetic Algorithms (GA) and Graph Theory.

GGC is an algorithm motivated to avoid the strong dependence between SC and its metric parameters, and in particular the Similarity Function that generates the Similarity Graph. Even though GGC takes an evolutionary approximation to clustering and uses some concepts from Graph Theory, it is strongly inspired by SC. This chapter describes in detail GGC and presents the two encodings and fitness functions that were studied in order to design the algorithm.

The algorithm first initializes the number of clusters, like in SC and K-means. Our technique looks for the best sub-graphs of the Similarity Graph which might define a clear partition. The Similarity Graph is generated by a Similarity Function like in the SC algorithm. The population is a set of potential solutions (named partitions) which evolve until a good solution is found, or a maximum number of generations are reached. The fitness function is a metric used to assess the potential solutions. The algorithm will try to maximize the fitness value. In the following, we describe the evolutionary components of GGC.

In order to consider the best way to generate the GA, it is important to study different aspects which are relevant during the algorithm design. First, we propose two possible encodings; second, we study the genetic operators; third, we focus our attention on the fitness function design. Once the algorithm has been designed, we generate a validation phase in order to choose the best design. Finally, we focus the study on several experiments in order to compare the new algorithm with some classical algorithms, comparing its results with synthetic and real-world datasets.

3.1 The GGC Encodings

The GA has been constructed using two classical integer encodings, well known in cluster-based genetic algorithms [71]. The first encoding is a simple vector encoding (label-based) while the second one is based on set theory (medoid-based). These two encodings have been selected to compare their computational effort and performance to choose the best encoding for our algorithm (the experimental comparison is shown in Section 3.4.1).

3.1.1 Label-based Encoding

We examined two encodings to choose the one with better performance. The first one follows the philosophy of what the literature named label-based [71] encoding. Each gene in the chromosome represent an x_i of the dataset, and its value indicates the cluster that it belongs to. This is a naïve encoding, genes contain an integer that identifies one cluster. The number of nodes in the graph determines the chromosome length. The Figure 3.1 shows an example of this encoding scheme with a chromosome containing the partition drawn in Figure 3.2. The following example illustrates this encoding:

Example 3.1.1. Let n be the number of data instances (in this case, we set $n = 9$). Let k be the number of clusters (we set $k = 3$). The chromosomes of Figure 3.1 shows three possible correct representations. The first chromosome of Figure 3.1, represents the clusters selection shown in Figure 3.2.

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	2	2	2	3	3	3
Chromosome 2	1	2	1	2	3	2	3	3	3
Chromosome 3	2	3	1	2	1	2	1	3	3

Figure 3.1: Label-based encoding in GGC algorithm.

3.1.2 Medoid-based Encoding

The second encoding is based on sets. In this case the chromosome is divided in several variable-length chunks, each one associated to a cluster. The chunks contain the data instances (medoids [71]) which compose each cluster. Probably, it can be better understood looking at Figure 3.3, whose first chromosome shows the partition of Figure 3.2 using this encoding. Each set represents a cluster and the elements of the sets are the data instances which compose each cluster.

Example 3.1.2. The chromosomes from the Example 3.1.1 can be represented with this new encoding is shown in Figure 3.3.

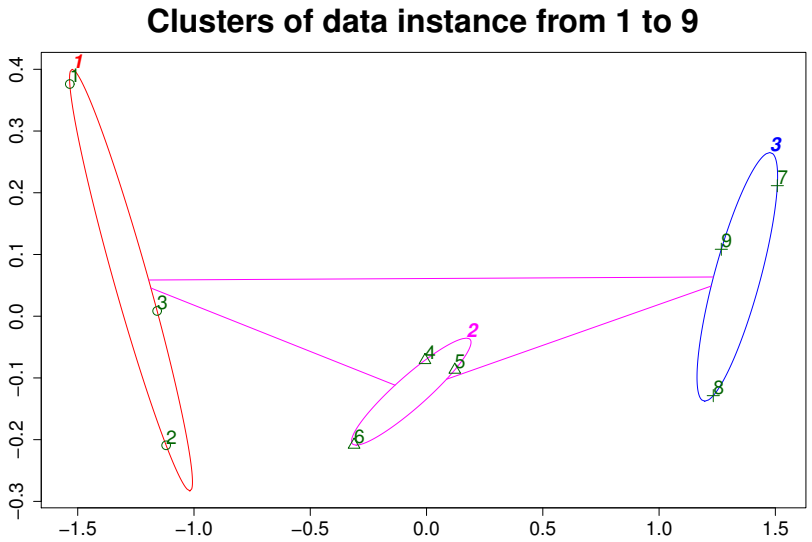


Figure 3.2: Representation of the clusters for the data instances of chromosome 1 of Figures 3.1 and 3.3.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 2, 3\}$	$\{4, 5, 6\}$	$\{7, 8, 9\}$
Chromosome 2	$\{1, 3\}$	$\{2, 4, 6\}$	$\{5, 7, 8, 9\}$
Chromosome 3	$\{3, 5, 7\}$	$\{1, 4, 6\}$	$\{2, 8, 9\}$

Figure 3.3: Medoid-based encoding of the GGC algorithm.

3.1.3 Invalid elements

The genetic operations (mutation and crossover) of the GA might create invalid chromosomes. Using previous encodings, it only happens when a chromosome contains one or more empty clusters. In partitional clustering, these solutions are invalid because the number of clusters is initially given, and therefore each cluster must contain at least one element. To avoid invalid chromosomes, the fitness value assigned to these chromosomes is 0. This value prevents that the elements passes to the next generation. Some examples of invalid elements for each encoding are shown in the following example.

Example 3.1.3. Figure 3.4 shows two chromosomes which are invalid elements for the label-based encoding, while Figure 3.5 shows the same elements using the medoid-based encoding and an invalid element which can only be generated using this encoding (the chromosome 3). If $k = 3$ and $n = 9$ the first individual has missed cluster 3 and the second cluster 1 (in both figures). In partitional clustering, all the clusters need to have at least one element. Chromosome 3 of Figure 3.5 repeats the assignation of one element (1) omitting other element (2).

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	1	1	2	2	2	2
Chromosome 2	2	2	2	3	3	3	2	2	2

Figure 3.4: Invalid chromosomes of the label-based encoding.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1,2,3,4,5\}$	$\{6,7,8,9\}$	$\emptyset$
Chromosome 2	$\emptyset$	$\{1,2,3,7,8,9\}$	$\{4,5,6\}$
Chromosome 3	$\{1,1,3,9\}$	$\{4,5,6\}$	$\{7,8\}$

Figure 3.5: Invalid chromosomes of the medoid-based encoding.

3.2 GGC Genetic Operators

This section describes the genetic operators which are used between the chromosomes for each encoding. The classical operators (selection, crossover and mutation) have been used.

3.2.1 Selection

Regardless of the encoding used, the selection operator selects a subset of chromosomes to reproduce and breed the offspring. These chromosomes are selected using a tournament [195]. In few words, a tournament selects randomly n chromosomes, assesses them using the fitness function, and then takes the fittest one. In this case, the fittest chromosomes of the generation pass to the next generation. It is called a $(\mu + \lambda)$ selection, where μ represents the number of bred chromosomes, and λ the new chromosomes generated.

3.2.2 Crossover

Any of the two encoding schemes induces a phenotype space smaller than the genotype space, and therefore different genotypes correspond to the same phenotype (see Figures 3.6, 3.7 and 3.8). This is a problem from the perspective of the recombination operator, because it destroys the correlation between phenotype and genotype spaces [175]. For this reason, it is recommendable to **relabel** the individuals before the application of the crossover. The criteria followed for this relabelling process is to maximize the similarity between the chromosomes which are crossed. It is focused on the convergence improvement of the algorithm by reducing the search space and the number of invalid elements. To this end we define the following similarity measure.

Definition 3.2.1 (Cluster Similarity measure)**.** Let $\{x_1, \ldots, x_n\}$ be a set of elements, and C_i,

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	2	2	2	3	3	3
Chromosome 2	2	2	2	3	3	3	1	1	1

Figure 3.6: Both chromosomes represent the same solution, but the name of the clusters appears different using the **label-based** encoding.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 2, 3\}$	$\{4, 5, 6\}$	$\{7, 8, 9\}$
Chromosome 2	$\{7, 8, 9\}$	$\{4, 5, 6\}$	$\{1, 2, 3\}$

Figure 3.7: Both chromosomes represent the same solution, but the name of the clusters appears different using the **medoid-based** encoding.

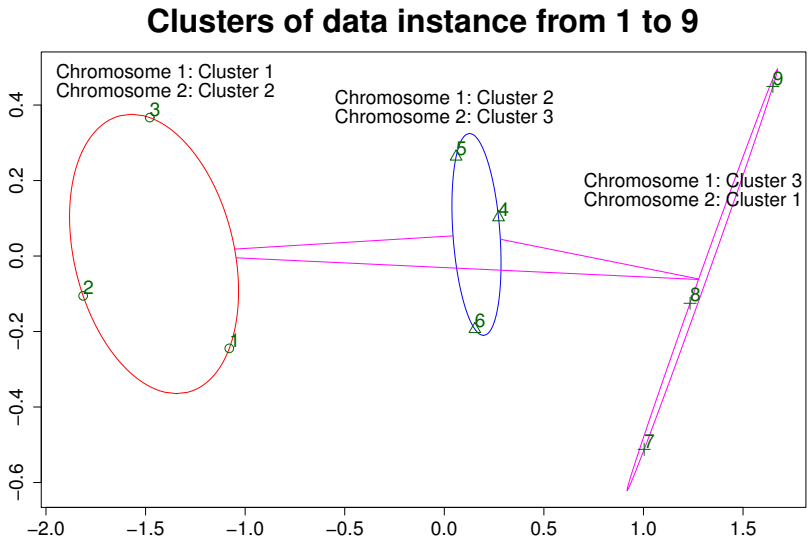


Figure 3.8: Cluster representation of chromosomes from Figures 3.6 and 3.7.

C_j the clusters which are compared. Their similarity measure is defined by:

$$sim(C_i, C_j) = \frac{1}{2}\left(\frac{\sum_{q=1}^{n} \delta_{C_i}(x_q)\delta_{C_j}(x_q)}{|C_i|} + \frac{\sum_{q=1}^{n} \delta_{C_i}(x_q)\delta_{C_j}(x_q)}{|C_j|}\right) \tag{3.1}$$

where $|C_i|$ is the number of elements of cluster C_i and $\delta_{C_i}(x_q)$ is the Kronecker δ defined by:

$$\delta_{C_i}(x_q) = \begin{cases} 0 & \text{if } x_q \notin C_i \\ 1 & \text{if } x_q \in C_i \end{cases}$$

The *relabelling process* can be divided in three fundamental steps:

1. The similarities between the clusters are calculated, using Equation (3.1).
2. The similarities are sorted using a decremental order.
3. The second chromosome is relabelled maximizing the similarity with the first chromosome.

Figures 3.9 and 3.10 show an example of two chromosomes, before the relabelling process for each encoding and the result of this process.

The crossover of the label-based encoding exchanges strings of numbers between two chromosomes. It is straightforward since both strings have the same length (see Figure 3.13). In the medoid-based encoding, it keeps the similar elements of both chromosomes and the different elements are randomly distributed amongst the clusters, creating two new elements (see Figure 3.14). Example 3.2.1 shows the crossover process for each encoding.

Example 3.2.1. This example shows the application of the relabelling process. First, it takes two chromosomes (see Figure 3.9 for the label-based encoding and Figure 3.10 for the medoid-based encoding) and calculates the similarities between the clusters. The results are shown in Table 3.1. Once the similarities are calculated, the clusters of chromosome 2 are relabelled using the most similar clusters:

- Cluster 3 is relabelled to 2 (similarity of 83,33%).
- Cluster 1 is relabelled to 3 (similarity of 66,67%).
- Cluster 2 is relabelled to 1 (similarity of 58,33%).

Figures 3.11 and 3.12 show the results of the relabelling process.

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	3	1	2	1	2	3	1	3
Chromosome 2	2	2	2	3	3	3	1	1	1

Figure 3.9: Example of two chromosomes for the relabelling process using the label-based encoding.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 3, 5, 8\}$	$\{4, 6\}$	$\{2, 7, 9\}$
Chromosome 2	$\{7, 8, 9\}$	$\{1, 2, 3\}$	$\{4, 5, 6\}$

Figure 3.10: Example of two chromosomes for the relabelling process using the medoid-based encoding.

Example 3.2.2. Figures 3.13 and 3.14 show the crossover process. The first figure exemplifies the label-based encoding. In this example, the exchange is between two sections of the chromosomes. These sections are randomly selected. In this case, the interval is from 4 to 7 (both numbers included). The second figure shows the crossover of the medoid-based encoding. In this case, the common parts of both chromosomes are kept and the rest of the data instances are randomly distributed by all the clusters.

Chromosome 1	Chromosome 2	Sim. Calculus	Sim. Percentage
Cluster 1	Cluster 1	$\frac{1}{2}\left(\frac{1}{4}+\frac{1}{3}\right)=\frac{7}{24}$	$29,17\%$
Cluster 1	Cluster 2	$\frac{1}{2}\left(\frac{2}{4}+\frac{2}{3}\right)=\frac{7}{12}$	**58,33%**
Cluster 1	Cluster 3	$\frac{1}{2}\left(\frac{1}{4}+\frac{1}{3}\right)=\frac{7}{24}$	$29,17\%$
Cluster 2	Cluster 1	$\frac{1}{2}\left(\frac{0}{2}+\frac{0}{3}\right)=0$	0%
Cluster 2	Cluster 2	$\frac{1}{2}\left(\frac{0}{2}+\frac{0}{3}\right)=0$	0%
Cluster 2	Cluster 3	$\frac{1}{2}\left(\frac{2}{2}+\frac{2}{3}\right)=\frac{5}{6}$	**83,33%**
Cluster 3	Cluster 1	$\frac{1}{2}\left(\frac{2}{3}+\frac{2}{3}\right)=\frac{2}{3}$	**66,67%**
Cluster 3	Cluster 2	$\frac{1}{2}\left(\frac{1}{3}+\frac{1}{3}\right)=\frac{1}{3}$	$33,33\%$
Cluster 3	Cluster 3	$\frac{1}{2}\left(\frac{0}{3}+\frac{0}{3}\right)=0$	0%

Table 3.1: Similarities from chromosomes shown in Figure 3.9.

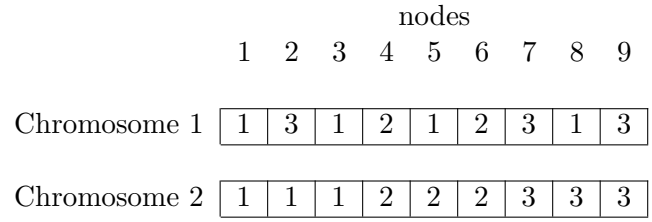


Figure 3.11: Example of the two chromosomes of Figure 3.9 after the relabelling process applied to chromosome 2 using the label-based encoding.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1,3,5,8\}$	$\{4,6\}$	$\{2,7,9\}$
Chromosome 2	$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$

Figure 3.12: Example of the two chromosomes from Figure 3.10 after the relabelling process applied to chromosome 2 using the medoid-based encoding.

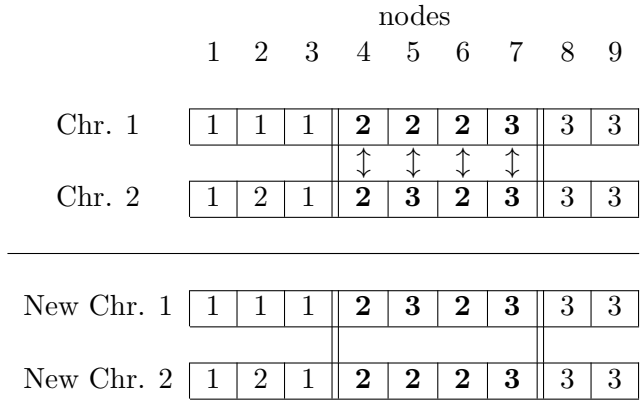


Figure 3.13: Crossover using the label-based encoding after relabelling.

3.2.3 Mutation

GGC uses an adaptive mutation for both encodings that works as follows:

	Clusters 1	Clusters 2	Clusters 3
Chr. 1	$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$
Chr. 2	$\{2,3\}$	$\{5,6,7\}$	$\{1,4,8,9\}$
Intersection	$\{2,3\}$	$\{5,6\}$	$\{8,9\}$
New Chr. 1	$\{2,3,\mathbf{4}\}$	$\{5,6,\mathbf{1}\}$	$\{8,9,\mathbf{7}\}$
New Chr. 2	$\{2,3,\mathbf{1},\mathbf{7}\}$	$\{5,6,\mathbf{4}\}$	$\{8,9\}$

Figure 3.14: Crossover using the medoid-based encoding.

1. A chromosome is randomly chosen to be mutated according to a mutation probability p_m, that is fixed at the beginning, with $p_m \in [0,1]$.
2. When a chromosome is chosen, the alleles which will be mutated are selected. The decision considers the probability of the allele to belong to the cluster which have been assigned. If the probability is high, the allele has a low probability of mutation and vice-versa. In our algorithm, this probability depends on the metric defined by the fitness function. This means that even if the mutation probability is high and an allele is chosen to mutate, if the chromosome is close to the solution it could not mutate.
3. Finally, the alleles are mutated depending on the encoding:
   - The label-based encoding changes the allele value. The new value is a random number between 1 and the number of clusters.
   - The medoid-based encoding moves the allele to other cluster. It randomly chooses the new cluster which will contain the allele.

Example 3.2.3. Figure 3.15 shows the mutation process for the label-based encoding. The first and seventh alleles have been randomly chosen to be changed. Figure 3.16 shows the same process applied to the medoid-based encoding: the third and eighth alleles have been moved from first and third clusters, to third and second respectively.

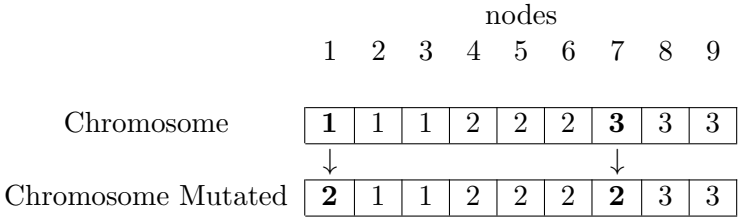


Figure 3.15: Mutation of two alleles in a chromosome using the label-based encoding.

3.3 The GGC Fitness Functions

This section describes the two fitness functions designed in the context of GGC; these functions have been chosen to satisfy the continuity condition of the clusters. The first fitness is the

	Cluster 1	Cluster 2	Cluster 3
Chromosome	$\{1, 2, \mathbf{3}\}$	$\{4, 5, 6\}$	$\{7, \mathbf{8}, 9\}$
Chromosome Mutated	$\{1, 2\}$	$\{4, 5, 6, \mathbf{8}\}$	$\{7, 9, \mathbf{3}\}$

Figure 3.16: Mutation of two alleles in a chromosome using the medoid-based encoding.

Weight Clustering Coefficient [20] which looks for "strong triangles" formed between neighbours in the graph. The second is based on a combination of the K-Nearest Neighbour [111] and the Mincut methods [177].

3.3.1 The Weighted Clustering Coefficient Fitness Function

The first fitness function uses Global Weight Clustering Coefficient [20] as the fit value for the population. Supposing an undirected weight graph, it applies the following metric C_i^w, which is defined as:

$$C_i^w = \frac{\sum_{j,h} \frac{w_{ij}+w_{ih}}{2} a_{ij} a_{ih} a_{jh}}{S_i(k_i - 1)} \tag{3.2}$$

where w_{ij} are the weights of the matrix, a_{ij} is 1 if the edge from i to j exists and 0 otherwise, $S_i = \sum_j w_{ij}$ and k_i is the number of neighbours of the node i. The denominator $S_i(k_i - 1)$ defines a normalization factor to range the value between $[0, 1]$. This fitness looks for individuals which have high similarity with their neighbours and whose neighbours also have high similarity between them.

3.3.2 KNN-Minimal Cut fitness

The second fitness function under study is a combination of the classical K-Nearest Neighbourhood (KNN) [111] and the Minimal Cut [177] algorithms. KNN is useful to guarantee the continuity condition which is frequent in the Spectral Clustering solutions. To control the separation between the elements of the clusters, the Minimal Cut measure is used. It guarantees that those elements which clearly belong to different clusters are not assigned to the same cluster. The K value for KNN is initially given by the user, nevertheless, in this work we have fixed it to 2 because it is the minimal value to guarantee the continuity, in a similar way than the Clustering Coefficient, and additionally it avoids over-fitting.

Algortihm 6 shows the pseudo-code of the fitness: KNN covers all the nodes and checks if the K-closest elements (related to the metric) are in the same cluster (lines 9 to 12). The fitness value of this measure is the mean of the percentage of well-classified neighbours of all the individuals in a cluster (lines 10 and 13). The Minimal Cut measure calculates the average value edge weights which have been removed (lines 11 and 14). The final value of the fitness is the product of the KNN metric and the subtraction between one and the Minimal Cut metric (line 16); both metrics have the same range: [0,1]. Therefore, the algorithm maximizes the value of

Algorithm 6 Pseudo-code of the KNN-Minimal Cut Fitness Function

Input: A n-vector of elements with values between 0 and k where k is the number of clusters and a variable *neighbours* which represents the number of neighbours for the KNN measure.
Output: A value between 0 and 1 which corresponds with the fitness achieved.

```
1: TotalKNN = 0;
2: TotalMC = 0;
3: Generate the set of k Clusters: C.
4: for all C_a ∈ C do
5:     if C_a = ∅ then
6:         return 0
7:     end if
8:     SumKNN = 0; SumMC = 0.
9:     for all ind ∈ C_a do
10:        SumKNN += PofKNN(neighbours, ind) {It calculates the percentage of neighbours for the individual ind which are assigned to the same cluster.}
11:        SumMC += AvEdWCut(ind) {It calculates the average value of the edge weights which have been cut from ind.}
12:    end for
13:    TotalKNN += SumKNN / |C_a|; {|C_a| represents the number of elements of C_a.}
14:    TotalMC += SumMC / |C_a|;
15: end for
16: return TotalKNN/|C| × (1 − TotalMC/|C|)
```

$\frac{TotalKNN}{|C|} \times \left(1 - \frac{TotalMC}{|C|}\right)$ where:

$$TotalMC = \sum_{x \in C} \frac{\sum_{y \notin C_x} w_{xy}}{|\{y | y \notin C_x\}|} \tag{3.3}$$

$$TotalKNN = \sum_{x \in C} \frac{|\{y | y \in \Gamma(x) \wedge y \in C_x\}|}{|\Gamma(x)|} \tag{3.4}$$

In these formulas, w_{xy} represents the weight of edge $x \to y$, C represents the set of clusters and $\Gamma(x)$ represents the neighbourhood of the element x. It reduces the weight values of the edges which are cut and improves the proximity of the neighbours.

3.3.3 The Algorithm Steps

The GGC algorithm can be divided in three main steps:

1. **Similarity Graph generation:** a Similarity Function (usually based on a kernel) is applied to the data instances (i.e., the domain concepts), connecting all the points with each other. It generates the Similarity Graph.

2. **Genetic search:** Giving an initial number of clusters k, the GA generates an initial population of possible solutions and evolves them using a fitness function to guide the

algorithm to find the best solution. It stops when a good solution is found, or a maximum number of generations is reached.

3. **Clustering association:** The solution with the highest fitness value is chosen as a solution of the algorithm and the data instances are assigned to the k clusters according to the solution chosen.

3.4 Algorithm Validation

This section shows an analysis of the GGC algorithm, including the two encodings introduced in Section 3.1 and the metrics associated with the fitness functions previously described. Finally, the robustness of the GGC algorithm is evaluated and compared against the robustness of the SC algorithm.

3.4.1 Comparison of GGC Encodings

The two encodings used in this work are equivalent and can be applied to any problem with similar results. However, they present the following differences:

- Omitting the relabelling process, the **label-based** crossover operation is faster than the medoid-based crossover. In the label-based case, the crossover is $O(n)$ because only one loop is necessary to swap the values of two vectors. For the medoid-based case, the crossover is $O(n^2)$ because two nested loops are necessary to find the common elements of two sets.
- The mutation effort of the two algorithms is almost the same, although the **label-based** encoding is slightly faster because in the label-based encoding the value changes instantly when the mutation is applied, while in the medoid-based encoding the value is extracted from one set and introduced in another set and a swap process is needed.
- Both encodings can use the relabelling process, however the **medoid-based** encoding simplifies the similarity calculus using the intersection operation.
- GGC algorithm presents, as any other heuristic-based search method, a local maximum convergence problem. This problem has not been deeply studied in the GGC algorithm, however it depends on the GA operators. To compare both encodings convergence behaviour, the Spirals dataset [101] has been tested against them. Figure 3.17 plots the convergence results for this dataset (for 50 runs of the algorithm per encoding and fitness function) using the parameters shown in Table 3.2, the KNN value set to 2 (as is explained in Section 3.3.2) and the tournament value also set to 2. The algorithm uses an adaptive mutation (see Section 3.2.3), the value (0.5) is the initial value for the mutation and (10^{-4}) the final value. In this case, the label-based encoding converges faster than the medoid-based encoding.

The **label-based** encoding reduces the computation effort (see Table 3.3). Therefore, it has been chosen to carry out the rest of the experiments.

Dataset	Pop.	Gen.	Cross.	Mut.	Eli.	Fit.
Spirals	200	2000	0.3	0.5-10^{-4}	50	1.0

Table 3.2: Parameter setting with population, generations, crossover probability, mutation probability and elitism size used with the Spirals datasets. The table also includes the fitness value achieved.

Encoding	Process			
	Crossover	Mutation	Relabelling	Convergence
Label-based	X	X		X
Medoid-based			X	

Table 3.3: Comparison for both encoding methods related to genetic operations in GGC. 'X' shows the encoding which achieves the best results with respect to computational effort and speed.

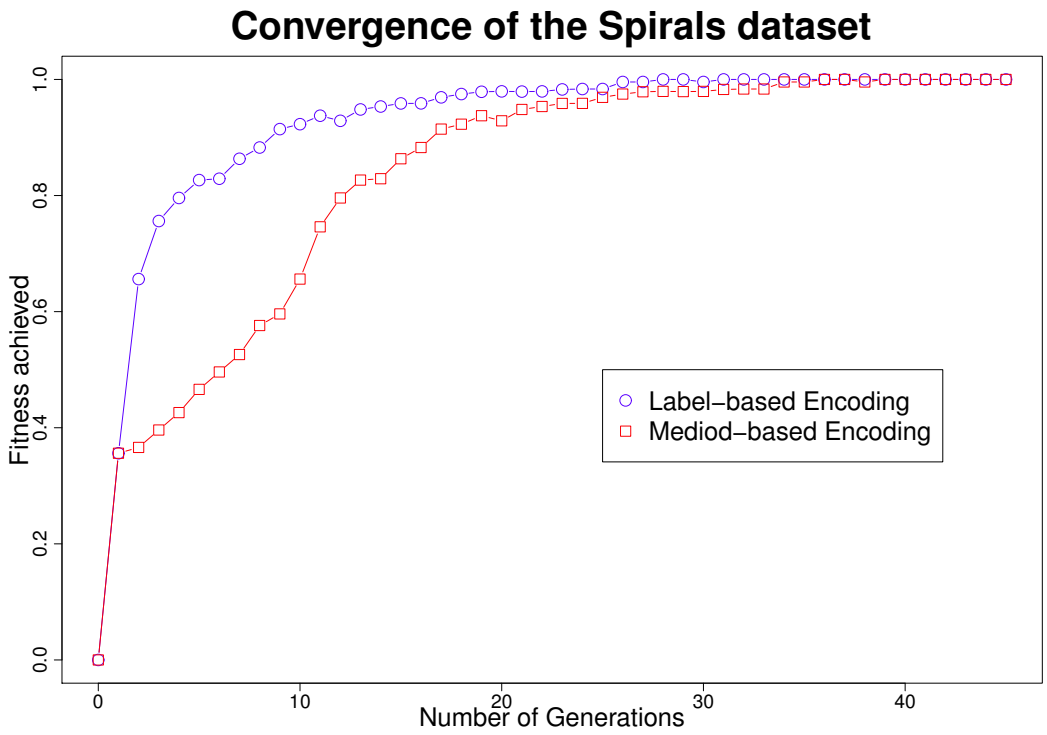


Figure 3.17: GGC convergence for Spirals dataset. The convergence for the label-based encoding is reached in the 30 generation, using the medoid-based encoding is reached in the 40 generation.

3.4.2 Fitness Functions comparison

During the experimental stage **to test the behaviour** of our fitness functions, we detected that the Weight Clustering Coefficient fitness obtained the maximum value even when the solution was incorrect. The analysis of this problem shows that it only happens when the Similarity Graph was fully connected (i.e., all the weights are bigger than 0). We analyzed this fact in more detail in an attempt to explain it, concluding that there is an issue with this approximation: It

can be mathematically proved that this problem is a "metric mistake" [1]. The following theorem shows the proof:

Theorem 3.4.1. *Suppose that G is a graph (with 3 elements or more) and W is the matrix of the weights of the graph. If $w_{ij} > 0\ \forall i, j$ then $C_i^w = 1\ \forall i$.*

Proof. We choose a random element i which has n neighbours. Let $x_1, \ldots, x_n$ be the weight values from the node i to its n neighbours. From the definition of the C_i^w we have:

$$C_i^w = \frac{\sum_{j,h} \frac{w_{ij}+w_{ih}}{2} a_{ij}a_{ih}a_{jh}}{S_i(k_i - 1)}$$

If we calculate S_i we have:

$$S_i = x_1 + \cdots + x_n$$

In this case $a_{ij} = 1\ \forall i, j$ and $k_i = n$, then:

$$C_i^w = \frac{\sum_{j,h} \frac{x_j+x_h}{2}}{S_i(n-1)}$$

If we sort the sum elements we have the following:

$$\begin{array}{ccccccc}
0 & + & \frac{x_1+x_2}{2} & + & \ldots & + & \frac{x_1+x_n}{2} \\
\frac{x_2+x_1}{2} & + & 0 & + & \ldots & + & \frac{x_2+x_n}{2} \\
\vdots & + & \vdots & + & \ldots & + & \vdots \\
\frac{x_n+x_1}{2} & + & \ldots & + & \frac{x_n+x_{n-1}}{2} & + & 0
\end{array}$$

If we consider the symmetries of the sum, and we sum the elements which are symmetric, then we have:

$$\begin{array}{ccccc}
(x_1 + x_2) + \cdots + (x_1 + x_n) & = & (n-1)x_1 & + & x_2 + \cdots + x_n \\
(x_2 + x_3) + \cdots + (x_2 + x_n) & = & (n-2)x_2 & + & x_3 + \cdots + x_n \\
(x_3 + x_4) + \cdots + (x_3 + x_n) & = & (n-3)x_3 & + & x_4 + \cdots + x_n \\
\vdots & = & \vdots & + & \vdots \\
(x_{n-1} + x_n) & = & (1)x_{n-1} & + & (1)x_n
\end{array}$$

In this case, if we sum, for example, the x_2 that is left in the first sum to $(n-2)x_2$ we have $(n-1)x_1$, if we do the same with the x_3 left in the first and second sum to $(n-3)x_3$ we have $(n-1)x_3$. If we continue until x_n we have $(n-1)x_i\ \forall i$. Then:

$$C_i^w = \frac{(n-1)(x_1 + \cdots + x_n)}{S_i(n-1)}$$

We know that $S_i = x_1 + \cdots + x_n$ then:

$$C_i^w = \frac{(n-1)(x_1 + \cdots + x_n)}{(x_1 + \cdots + x_n)(n-1)} = 1$$

□

[1] This metric has also been used in several works about Weighted Complex Networks [20] and it is an important reference in the literature.

Since the Similarity Graph construction that was chosen is the fully connected graph (see Section 2.4.3), the only fitness that has been applied in the experiments is the KNN-Minimal Cut fitness to avoid this problem.The fully connected approximation was chosen because the GGC algorithm tries to maximize the robustness of the clustering selection related to the metrics, as is explained in the following subsection. Therefore, if the ϵ-neighbourhood graph or the k-nearest neighbour graph are chosen (see Section 2.4.3), the Similarity Graph increments the number of zero similarities, which is not desirable when all the elements could have a non-zero similarity between them. It could reduce the robustness of the algorithm and supposes a higher dependency on parameters; in this case, the Similarity Graph generation parameters: the ϵ value of the ϵ-neighbourhood graph, and the k value of the k-nearest neighbour graph.

3.4.3 Metrics

All the techniques use the metrics which have been mentioned before: K-means and EM use the Euclidean Distance Metric and Spectral Clustering and GGC use the Radian Basis Function (with the σ parameter optimized in the Spectral Clustering case). There are not applications with other metrics (or kernels) because the goal of the GGC algorithm is to be robust enough to separate the dataset without orders of magnitude problems. It tries to give the same results if two different metrics give the same relative distance between the objects, that is, if there are, for example, three objects: o_1, o_2, o_3, and the distances between them are $d_{12}^{M_1}$, $d_{23}^{M_1}$, $d_{13}^{M_1}$, $d_{12}^{M_2}$, $d_{23}^{M_2}$, $d_{13}^{M_2}$ where M_1 and M_2 are the metrics and the two metric distances satisfies the same order relationship:

$$d_{12}^{M_1} < d_{23}^{M_1} < d_{13}^{M_1}$$

$$d_{12}^{M_2} < d_{23}^{M_2} < d_{13}^{M_2}$$

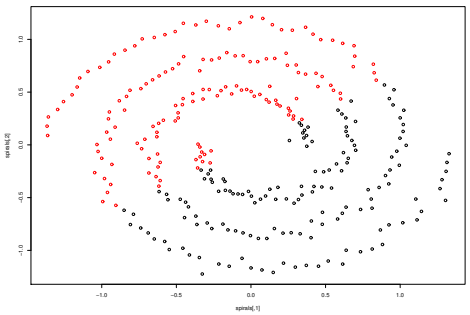
Then the clustering results should be almost the same (except, for example, when some of these distances are infinity or zero).

3.4.4 Comparing SC and GGC Robustness

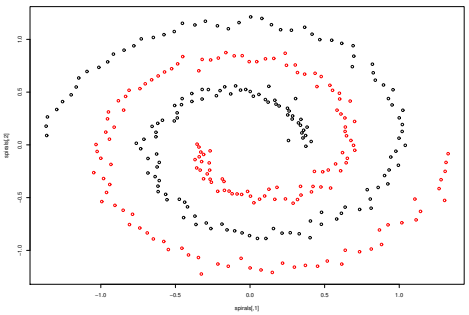
An important problem related to SC is related to its dependency on the parameters of the Similarity Function. The GGC algorithm has been designed to alleviate this problem. The KNN metric which is applied in the fitness calculation provides a higher robustness to the algorithm compared to the SC algorithm, it does not depend on the order of distance magnitude calculated by the metric. Figure 3.18 shows a clear example. In this case, the Spectral Clustering algorithm (implemented in the "kernlab" package of CRAN [101]) is compared against the GGC algorithm. In the "kernlab" package, Karatzoglou et al. implements the Random Walks Normalized Spectral Clustering algorithm. They use the Gaussian RBF Kernel to set the Similarity Graph. It is defined by:

$$K_{ij} = e^{-\sigma||x_i - x_j||^2} \tag{3.5}$$

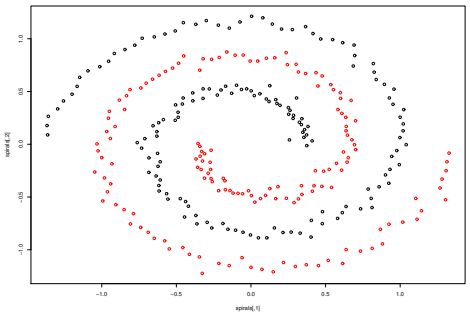
Where K is the Similarity Graph, x_i, x_j are data instances, and σ is the parameter which changes the order of magnitude. The experimental results show that the clustering technique clearly depends on the σ parameter. Figure 3.19 shows the different clustering results obtained using the SC and the GGC algorithms modifying the σ parameter between 1 and 4000.



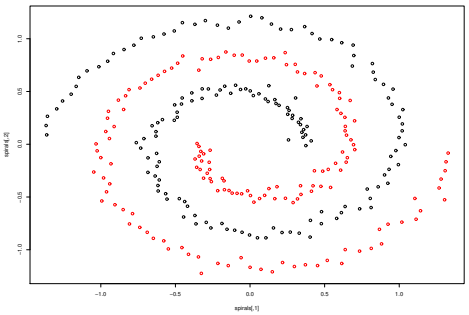
(a) Spectral Clustering results of Spirals for $\sigma = 2$



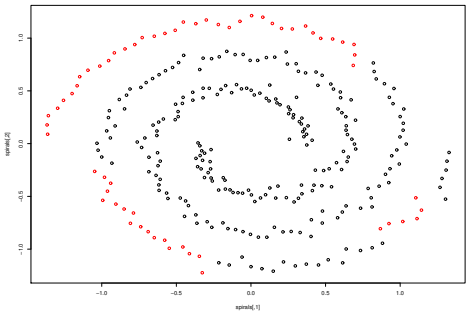
(b) GGC results of Spirals for $\sigma = 2$



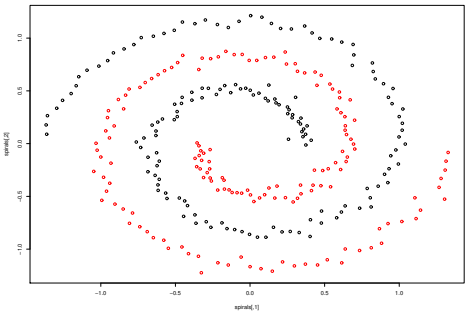
(c) Spectral Clustering results of Spirals for $\sigma = 500$



(d) GGC results of Spirals for $\sigma = 500$



(e) Spectral Clustering results of Spirals for $\sigma = 2000$



(f) GGC results of Spirals for $\sigma = 2000$

Figure 3.18: Spectral Clustering and GGC results for the Spirals [101] dataset with $\sigma = 2$, $\sigma = 500$, $\sigma = 2000$, respectively.

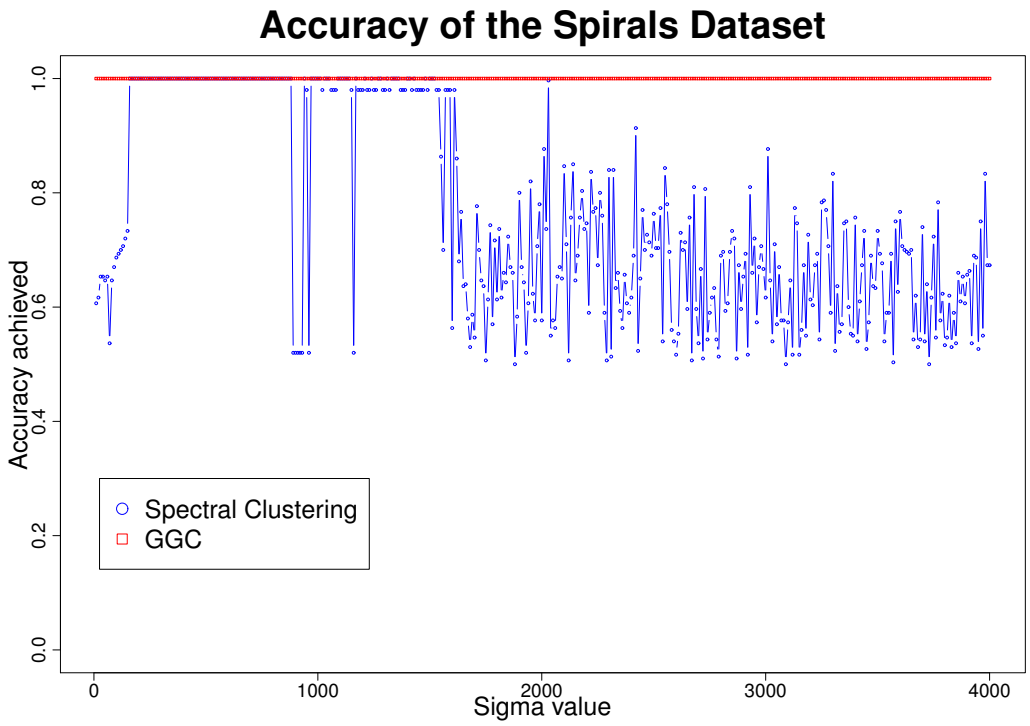


Figure 3.19: SC and GGC results for the Spirals [101] dataset with σ values from 1 to 4000, respectively. The red straight line in the top represents the robustness of GGC over SC.

These experimental results show that the parameters used in the definition of the kernel are critical (see the evolution of σ in Figure 3.19) because these parameters define the degree of the similarity. Ng et al. introduced a method to calculate the optimal σ in [155], however, as Figure 3.22 shows, this technique is not always enough. GGC always obtains the same results because it has been designed to be robust to the modification of the metric parameters, when this modification keeps the order relationship between the elements of the dataset and only changes the distance magnitude. The next section will show the experiments carried out using the GGC algorithm. The accuracy of the algorithm has been tested using synthetic and real datasets.

3.5 Experimental Results

This section compares the GGC algorithm with other classical clustering algorithms (K-means, EM and SC) using synthetic and real datasets. The accuracy value is calculated using the similarity metric defined in Equation 3.1.

3.5.1 Experiments on Synthetic data

Eight datasets have been extracted from the state of the art in clustering research area which study the behaviour of different algorithms similar to SC [37, 73, 78, 96, 192, 208].

Data	Instances	Clusters	Structure
Ag	788	7	Parametric
Cp	399	6	Mixture
D31	3100	31	Noisy Parametric
Fl	241	3	Continuity
Jn	373	2	Continuity
Pb	300	3	Noisy Continuity
R15	600	15	Parametric
Sp	312	3	Continuity

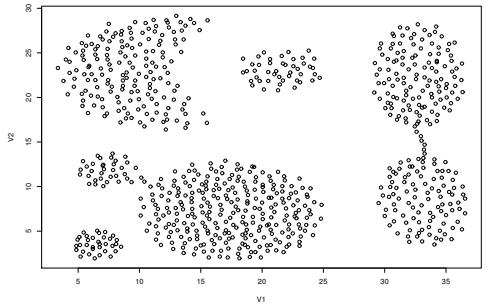
Table 3.4: Synthetic datasets, and their basic features, used to evaluate the GGC algorithm performance.

3.5.1.1 Data description

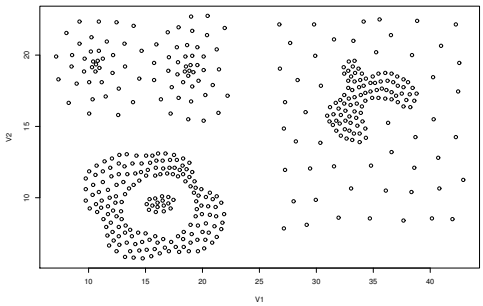
The initial datasets considered are 2-dimensional data which can be separated by human intuition, but are problematic to classical clustering algorithms. We have analysed the following datasets (see Figure 3.20 for a graphical representation):

- Aggregation [78] (Ag): This dataset is composed by 7 clusters, some of them can be separated by parametric clustering methods (see Figure 3.20 (a)).
- Compound [208] (Cp): There are 6 clusters which are only separable by non-parametric methods (or using special kernels if parametric clustering is applied), as it is shown in Figure 3.20 (b).
- D31 [192]: This data has 31 clusters with a high level of noise (see Figure 3.20 (c)).
- Flame [73] (Fl): This dataset has three ideal clusters: the first one is the base of the figure, the second one is the top and the last one are three outliers at the top-left of the image (see Figure 3.20 (d)).
- Jain [96] (Jn): This dataset is composed of two surfaces with different density and a clear separation (see Figure 3.20 (e)).
- PathBased [37] (Pb): This dataset has 2 clusters which can be separated by a parametric method and another cluster which can only be separated by a non-parametric method. This example is problematic for algorithms such as Spectral Clustering because this algorithm is sensitive to noisy data (see Figure 3.20 (f)).
- R15 [192]: Similar to D31, this dataset is divided in 15 clusters which are clearly separated (see Figure 3.20 (g)).
- Spiral [37] (Sp): In this case, there are 3 spirals close to each other (see Figure 3.20 (h)).

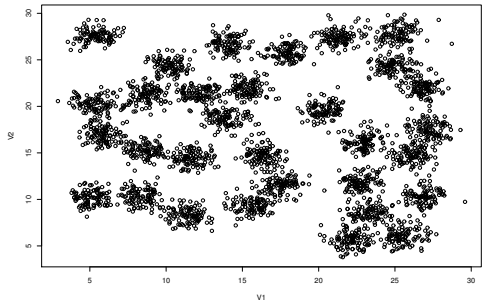
Finally, Table 3.4 summarizes the features of the datasets.



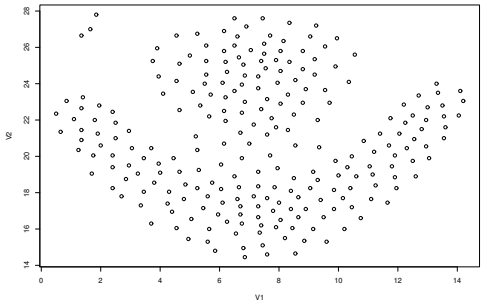
(a) Original Aggregation dataset



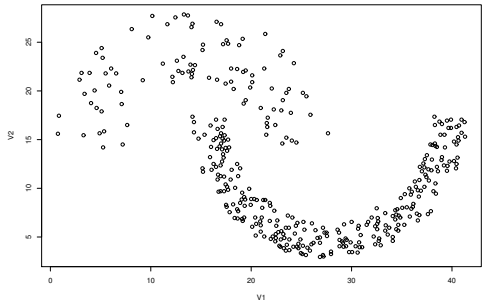
(b) Original Compound dataset



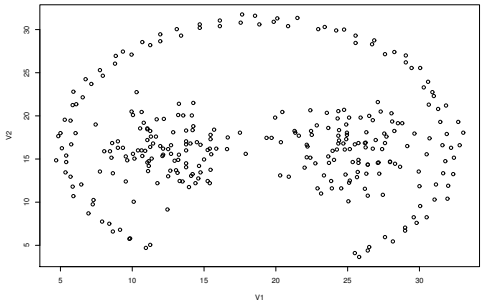
(c) Original D31 dataset



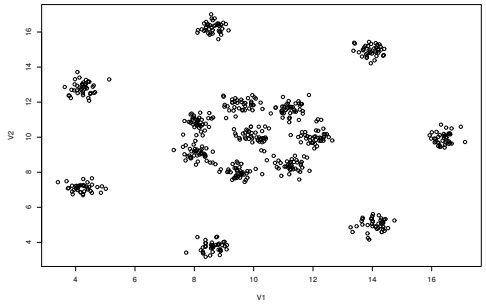
(d) Original Flame dataset



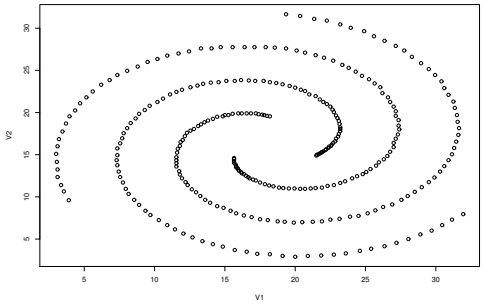
(e) Original Jain dataset



(f) Original PathBased dataset

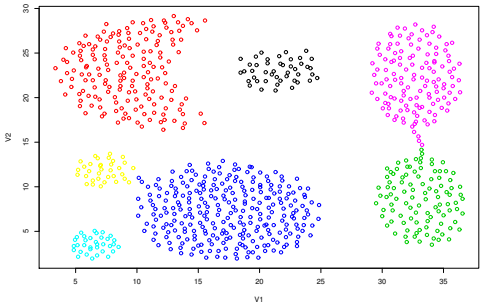


(g) Original R15 dataset

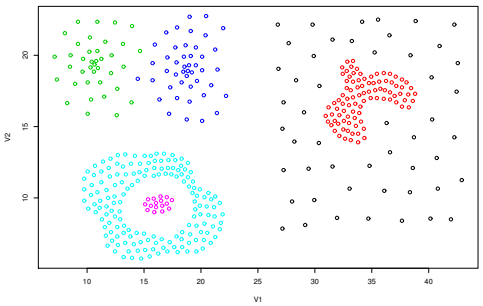


(h) Original Spiral dataset

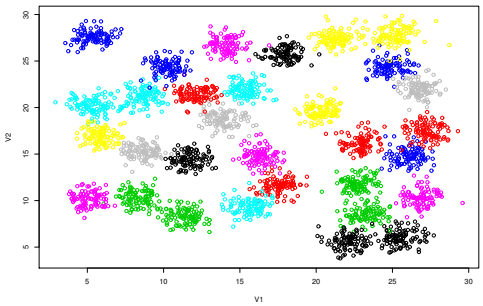
Figure 3.20: The original images of the synthetic datasets.



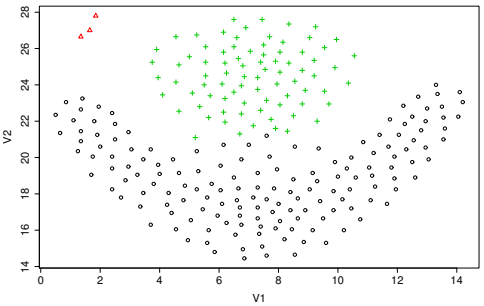
(a) Ideal human separation for Aggregation



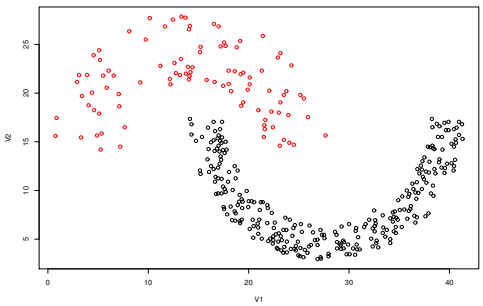
(b) Ideal human separation for Compound



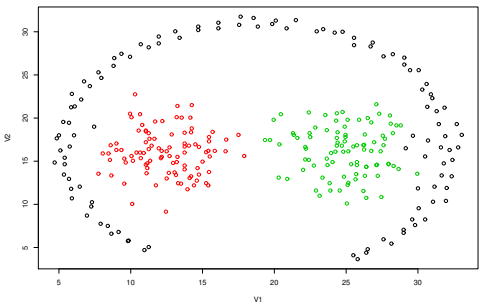
(c) Ideal human separation for D31



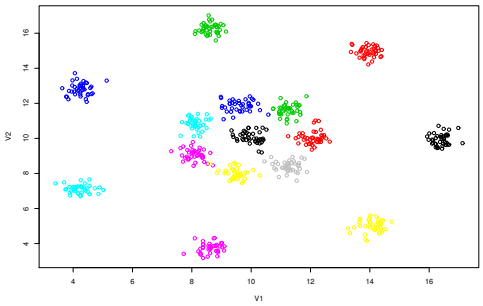
(d) Ideal human separation for Flame



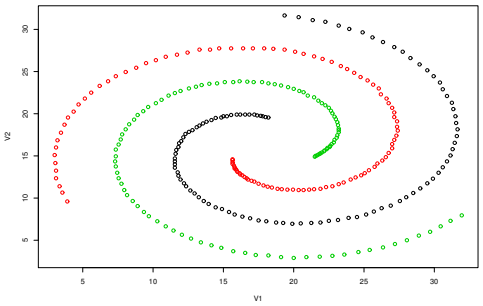
(e) Ideal human separation for Jain



(f) Ideal human separation for PathBased

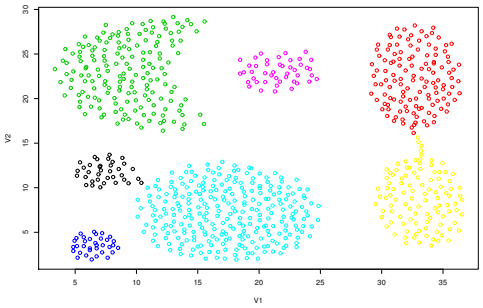


(g) Ideal human separation for R15

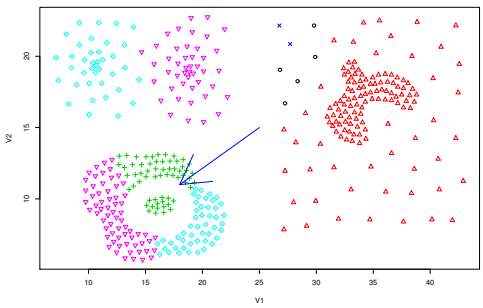


(h) Ideal human separation for Spiral

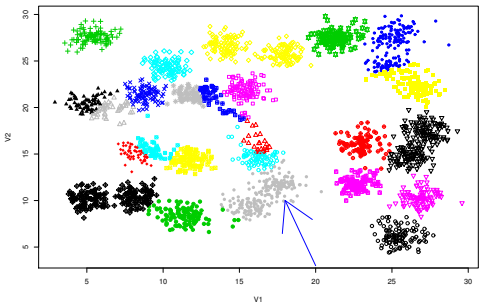
Figure 3.21: The ideal human-based separation of the synthetic datasets.



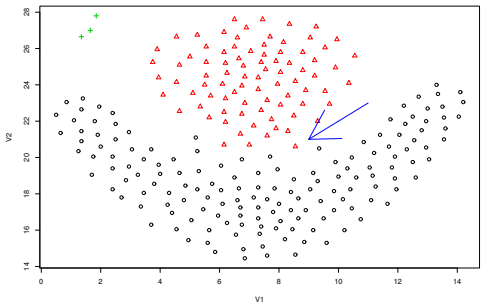
(a) SC results for Aggregation



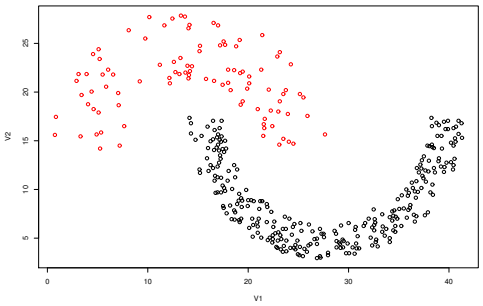
(b) SC results for Compound



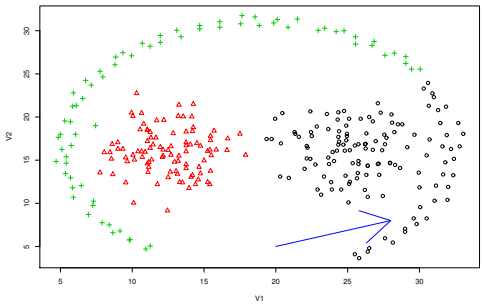
(c) SC results for D31



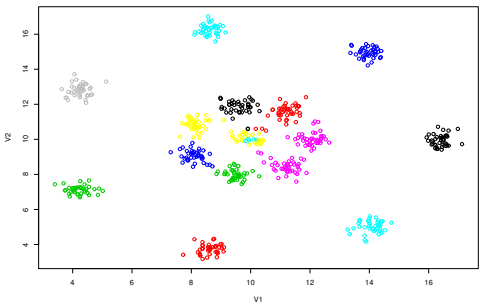
(d) SC results for Flame



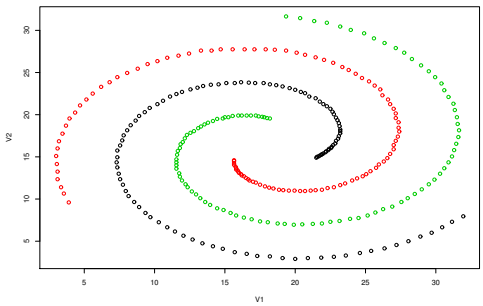
(e) SC results for Jain



(f) SC results for PathBased

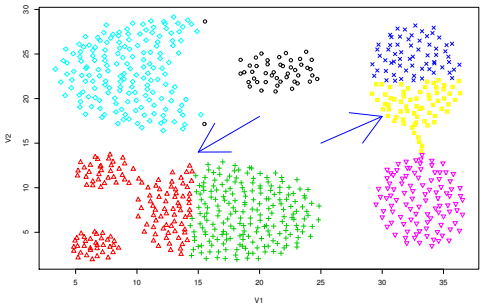


(g) SC results for R15

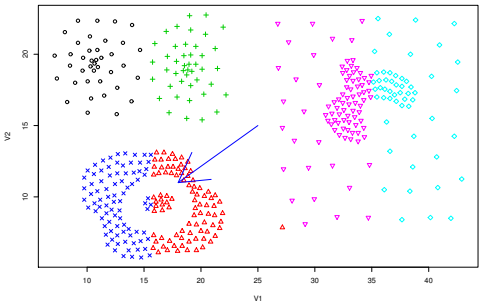


(h) SC results for Spiral

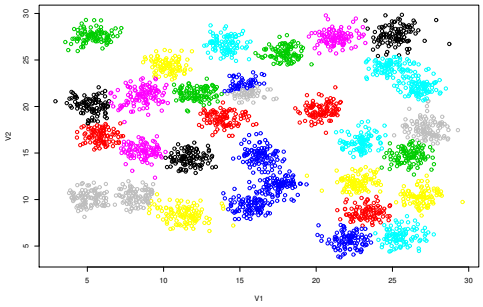
Figure 3.22: Spectral Clustering results for the synthetic datasets. The algorithm has problems with Compound, D31, Flame, PathBased and R15.



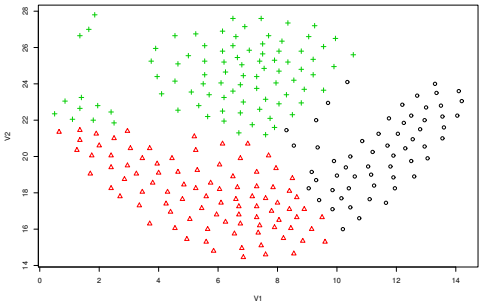
(a) K-means results for Aggregation



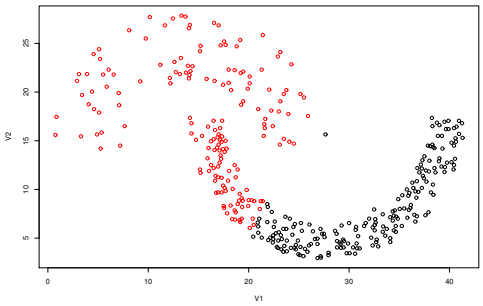
(b) K-means results for Compound



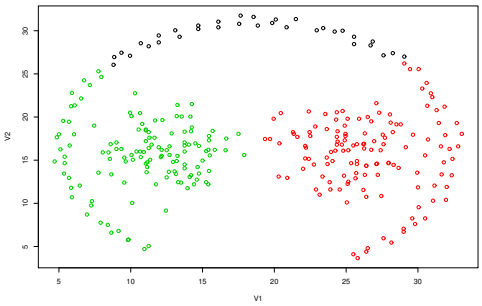
(c) K-means results for D31



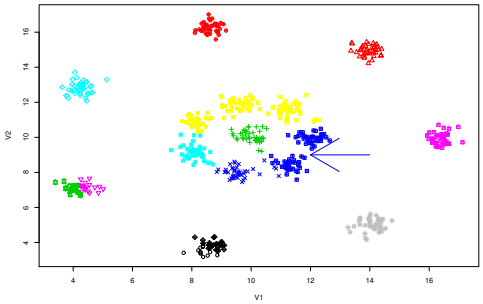
(d) K-means results for Flame



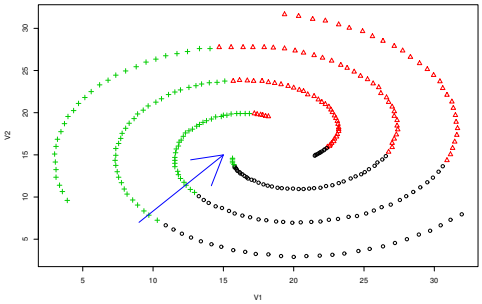
(e) K-means results for Jain



(f) K-means results for PathBased

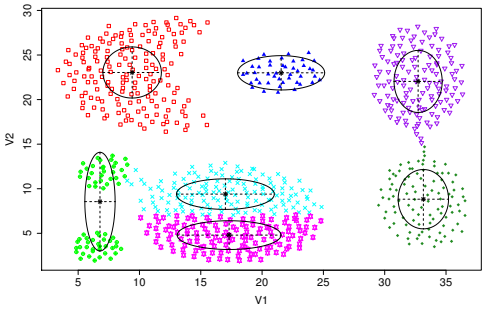


(g) K-means results for R15

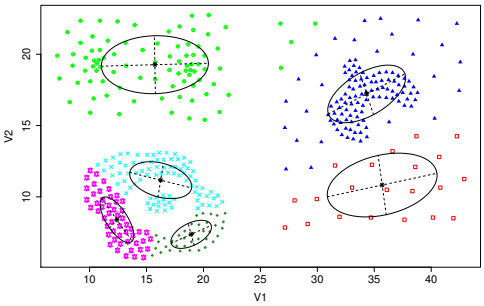


(h) K-means results for Spiral

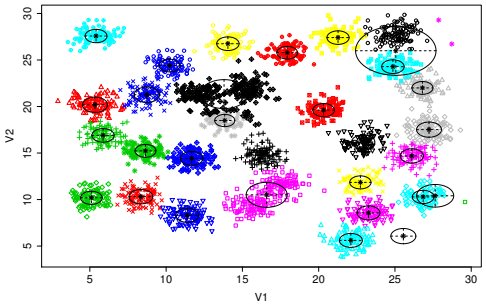
Figure 3.23: K-means results for the synthetic datasets. The algorithm has problems with all the datasets.



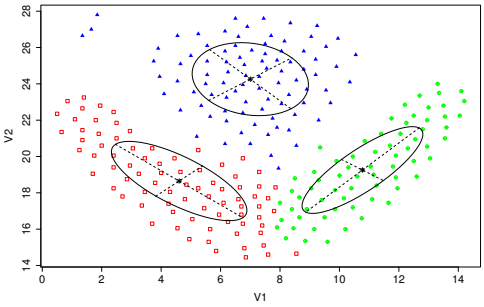
(a) EM results for Aggregation



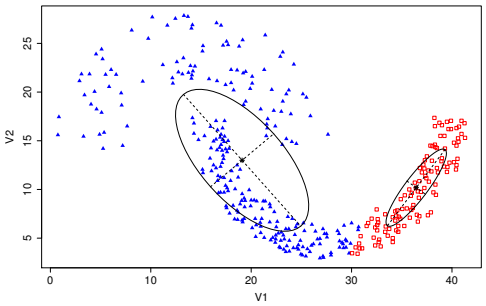
(b) EM results for Compound



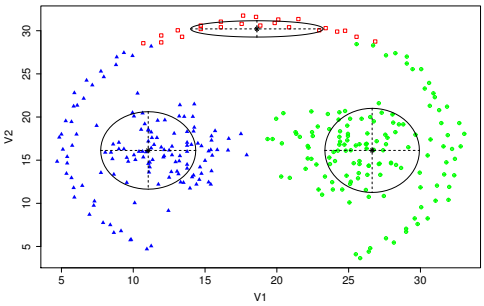
(c) EM results for D31



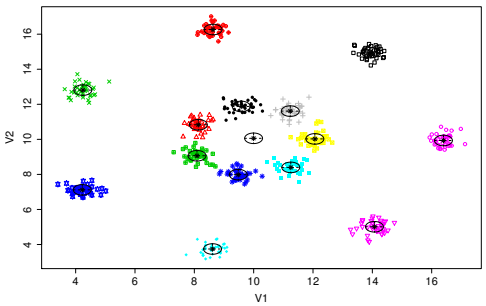
(d) EM results for Flame



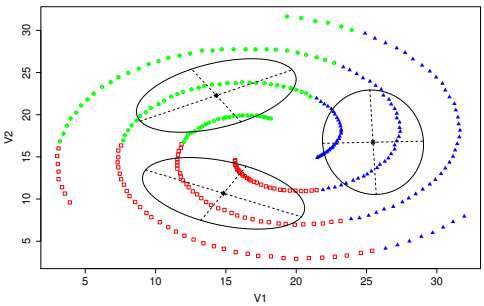
(e) EM results for Jain



(f) EM results for PathBased

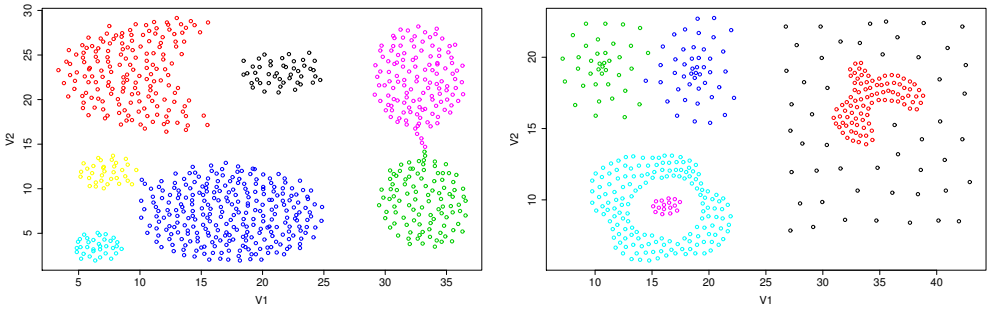


(g) EM results for R15



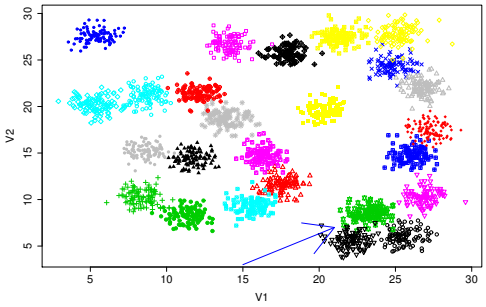
(h) EM results for Spiral

Figure 3.24: EM results for the synthetic datasets. The algorithm has problems with all the datasets except R15.

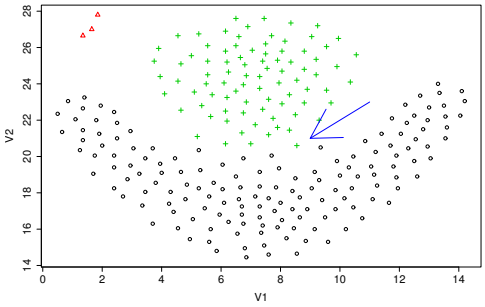


(a) GGC results for Aggregation

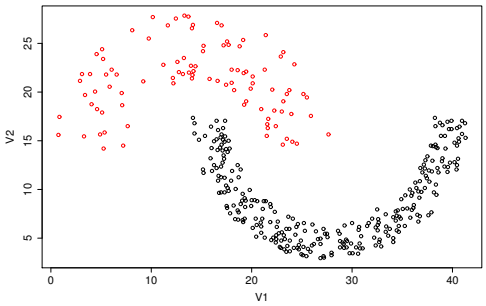
(b) GGC results for Compound



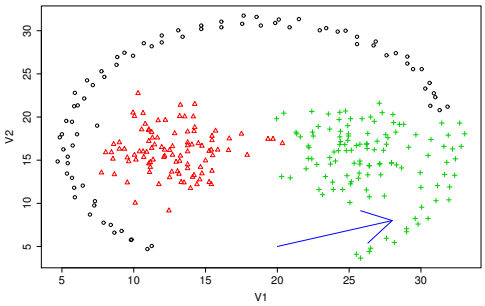
(c) GGC results for D31



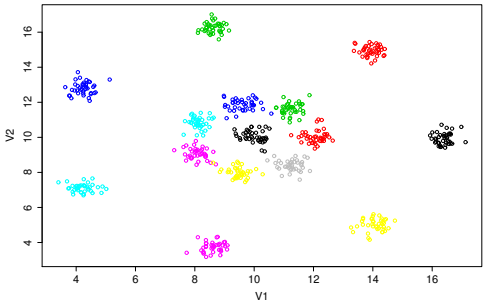
(d) GGC results for Flame



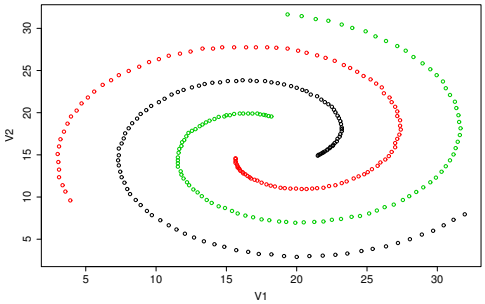
(e) GGC results for Jain



(f) GGC results for PathBased



(g) GGC results for R15



(h) GGC results for Spiral

Figure 3.25: GGC results for the synthetic datasets. The algorithm has problems with Flame and PathBased.

Data	SC	GGC	EM	K-Means
Pb	**89%**	*88%*	71%	74%
Ag	*96%*	**▲100%**	79%	86%
D31	85%	**▲99%**	*90%*	82%
Cp	*77%*	**▲100%**	57%	72%
R15	81%	**▲100%**	*100%*	81%
Jn	**100%**	**100%**	57%	78%
Sp	**100%**	**100%**	35%	35%
Fl	**99%**	*99%*	69%	70%

Table 3.5: Results for the different datasets applying K-means, Expectation Maximization, Spectral Clustering and the GGC algorithm. The best results have been marked in bold and the second best in italic. Statistical significant improvements are indicated by a ▲ symbol.

Comparison	Obs. Dif.	Crit. Dif	Differente
SC–GGC	5	13.6239	FALSE
SC–EM	11	13.6239	FALSE
SC–K-Means	10	13.6239	FALSE
GCC–EM	16	13.6239	TRUE
GCC–K-means	15	13.6239	TRUE
EM–K-means	1	13.6239	FALSE

Table 3.6: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets.

3.5.1.2 Experimental Results

The selected clustering algorithms (K-means, EM using a Gaussian Mixture Model estimator, SC and the GGC algorithm) have been applied to the previous described datasets. We carried out an experiment executing the algorithms 50 times and taking their best results. Figure 3.20 shows the original datasets, whereas Figure 3.21 shows the ideal clusters. We selected best fitness as performance measure due to two main reasons. First, the goal is to maximize the fitness to achieve the best cluster discrimination, i.e., what Eiben and Jelasity named design domain, and therefore the best fitness is the better choice [61]. Secondly, in our experiments, and on the contrary than other authors observations [124], we observed that similar fitness values are associated to quite different genotypes. It follows the same reasoning of [61] where Eiben and Jelasity explain when these two approaches should be used in GA. Table 3.5 shows the best accuracy results, and Table 3.7 shows the parameters and the best fitness values achieved by the GGC algorithm for these datasets. GGC and SC use the RBF kernel [77]. EM and K-means use the Euclidean distance [56].

Table 3.5 and Figure 3.22 show that Aggregation, Jain and Spirals are not problematic for SC (we are using the Ng [155] version of the algorithm). However, Compound, Flame, PathBased, D31 and R15 are more problematic. Compound is difficult to classify using SC because the distribution of the data is highly heterogeneous. In the case of Flame, there is not a clear boundary between the clusters. It makes difficult the application of the algorithm. D31 and PathBased have noisy information (see Figure 3.22), it produces several deviations for the SC algorithm. R15 has also noisy information in the central clusters.

Figure 3.23 shows the results of Kmeans using the Euclidean distance metric. K-means, as a parametric technique, does not obtain good general results. The reason is that the parameter is a set of centroids optimized by the algorithm. In the case of Compound, for example, the clusters of the top-left position of the image (see Figure 3.23) are well classified, however it is impossible, with these conditions, that the algorithm classifies correctly the bottom-left two clusters because one cluster surrounds the other (see Figure 3.23). The same problem appears with Jain, Spirals, PathBased and Flame. In the case of Aggregation, the worst misclassification is related to the three clusters of the bottom-left and the two clusters of the right. In this case, the different sizes of the clusters influence the selection process. The D31 and R15 misclassification might be a consequence of a local minimum convergence of the algorithm caused by the noisy information.

EM obtains better results than K-means but it also has problems with other datasets. It achieves better results for R15 although the rest of the datasets are misclassified (see Figure 3.24).

Finally, the GGC algorithm achieves good results in almost all the cases (see Figure 3.25). Table 3.7 shows the parameters selection of the GA for each case. The results show that the GGC algorithm only has problems with the most noisy cases: Flame, Pathbased and D31. The reason is related to a boundary problem. It is difficult for the algorithm (using the RBF metric in the generation of the Similarity Graph), to determine the limits of the clusters when they are not clear. Also, even if the algorithm has achieved the maximum accuracy values, there are some cases where the fitness function does not obtain the maximum value of its range. It is usual that hard problems such as Compound or D31 do not permit the fitness to find a max-range solution, even if the final cluster selection achieved by the algorithm is closed to the human selection. Applying the Friedman test [89] to the four algorithms, the p value achieved is 0.002037, which means that, at least, two of the algorithms are significantly different from each other. Table 3.6 shows the multiple comparisons related to the Friedman test. The algorithms which are significantly different are GGC compared with K-means and EM. Due to GGC has no statistical difference with SC, we have compared the values using a Wilcoxon Test [201] per dataset (see triangles in Table 3.5), we can see that there is significantly difference in, at least, 4 cases.

Data	**Pop.**	**Gen.**	**Cross.**	**Mut.**	**Sel.**	**Max. Fitness**
Ag	100	2000	0.4	0.01	50	0.9928
Cp	200	2000	0.5	0.01	50	0.9552
Fl	100	2000	0.4	0.01	50	0.9828
Jn	100	500	0.4	0.2	50	1.0
Pb	100	2000	0.4	0.01	50	1.0
R15	200	2000	0.5	0.3	50	0.9850
Sp	100	500	0.4	0.01	50	1.0
D31	200	5000	0.7	0.4	50	0.9445

Table 3.7: Best parameter selection found of the GGC algorithm for the different synthetic datasets and the fitness achieved by the GGC algorithm. The K value of the KNN-Minimal Cut fitness is always set to 2.

3.5.2 Experiments on Real-World data

Finally, some experiments have been focused on real datasets which have been previously classified by humans.

3.5.2.1 Dataset Description

The experiments have been applied on three real datasets extracted from the UCI Machine Learning Repository [70]:

- **Iris** (Ir): This dataset is a well-known dataset. It has 150 instances of 3 different classes (50 per class). Each class refers to a type of iris plant: Iris Setosa, Iris Versicolour and Iris Virginica. Each instance has 4 attributes which are Sepal length, Sepal width, Petal length and Petal width. It does not have missing values.
- **Wine** (Wn): This dataset has 178 instances. Each instance has 13 attributes and can belongs to 1 of the three different classes. Each class refers to a type of wine. The first class has 59 instances, the second one has 71 and the third one has 48. It does not have missing values.
- **Handwriting** (HW): This dataset is based on digits handwriting (see Figure 3.26). It has 60000 train instances and 10000 test instances. Each instance has a vector of 784 elements which represents a 28x28 matrix where each element is a pixel in grayscale ranged from 0 to 256. There is also a column for the labels numbered from 0 to 9. It does not have missing values. In this work only 6000 instances of this dataset has been analysed because the Similarity Graph generated by the Spectral Clustering algorithm is bigger than the memory available [2].

3.5.2.2 Preprocessing and Normalizing the Data

The preprocessing process is divided in two main steps:

- The first step has been the study of the available variables through histograms and correlation diagrams which were used for dimension reduction. The information provided by this phase shows the values which are useless because, for example, are constants or have a high correlation (more than 0.8 if we consider that the correlation values is in range $[0, 1]$) with other variables. This means that they may variate the clustering results, if they are not eliminated, with redundant information.
- The second preprocessing phase consists on the normalization of the variables. First, the attributes with outliers are recentralized. After, the same range is applied for all. As

[2] The computer used has 4 Gbytes of RAM memory and 1 Gbytes of Virtual Memory, in the generation of the Similarity Graph it is necessary to generate a matrix of 6000×6000 of double values. If a double variable requires 8 bytes, then the whole matrix requires $6000 \times 6000 \times 8 \approx 288$ Mbytes. However, if the 60000 data instance are used, the memory required is $60000 \times 60000 \times 8 \approx 28.8$ Gbytes.

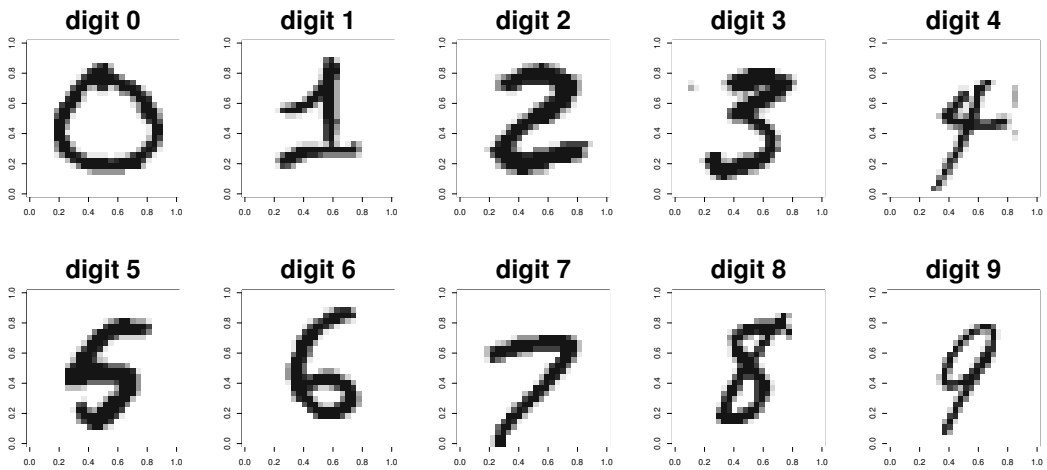


Figure 3.26: Example of the digits dataset

was described in Section 2.3, we combine Z-score [36] to recentralized the distribution and avoid outliers and MinMax [84] to fixed the range of all the values between 0 and 1.

Figures 3.27 and 3.28 show the boxplots and histograms of the Iris dataset for all its variables and classes. In the Iris and Wine datasets, there are a few number of instances and attributes, it implies that the reduction is not necessary. However, in the case of the handwriting dataset there are a lot of attributes (pixels) which do not contribute to the analysis (those pixels which have always the same value, for example). Also there are features which have a high correlation between them. These attributes have been reduced in the first step leaving 195 of 784 attributes for the analysis. All the attributes of the datasets have been normalized applying the techniques of the second step.

3.5.2.3 Experiment Results

The experiments have followed the same procedure that they followed with the synthetic datasets experiments (see Table 3.8 for the parameters selection). The value of σ has been approximated to 100 using the same methodology that was used in the synthetic analysis. Table 3.9 shows the accuracy percentages of the different clustering algorithms. The results for the Iris show that EM is the best classifier (with an accuracy of the 96,67 %) and the GGC algorithm is the second (92%). The results for the Wine datasets show that all the algorithms obtained high accuracy values (bigger than the 95 %), and the GGC algorithm obtains a perfect classification with the maximum fitness value (see Table 3.8). Finally, the results of the Handwriting show that Spectral Clustering and GGC obtain the best classification results (73,55% and 99%, respectively).

These results are a consequence of the data distribution. Iris dataset has instances of different classes which are close to each other (see Figure 3.29); the GGC algorithm has problems to discriminate the boundary of the clusters specially when there are intersections between the clusters. The fitness value of the Iris is the highest that the algorithm has achieved, it shows that there are instances which belongs to different clusters but are close to each other. In the case of the Wine dataset, the classes are clearly separated, as the different clustering techniques

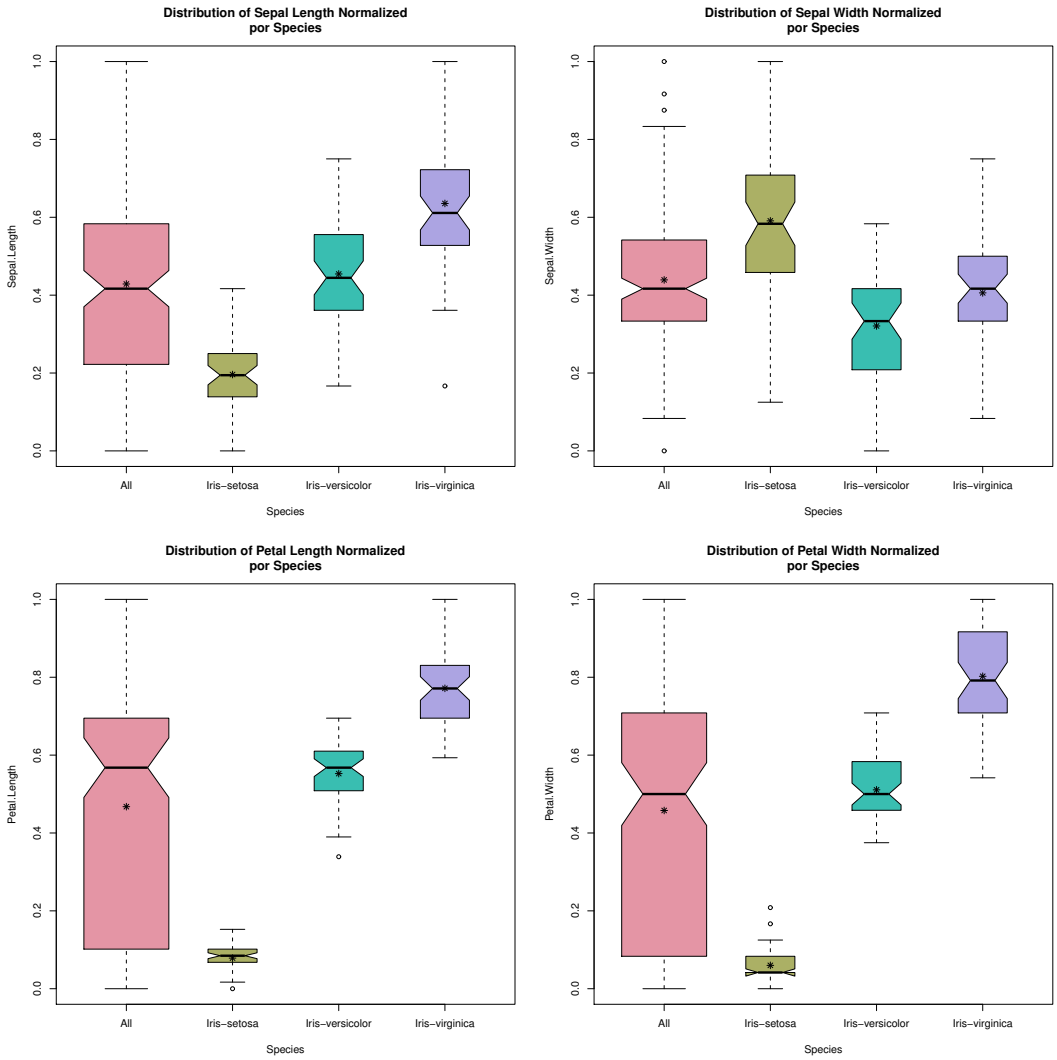


Figure 3.27: Boxplot of the Iris Dataset. Petal-length and Petal-width are the variables that better discriminates the three classes. Versicolour and Virginica classes are more difficult to discriminate than Setosa class. The number of outliers is low.

show. It improves the results of the GGC algorithm, because the boundaries are clearer. It must be also similar in the Handwriting case, however, the fitness value shows that there are some instances in the cluster boundaries and they are difficult to assign. The p-value obtained applying the Friedman test is 0.2123 which means that there is not significantly difference among the algorithms. However, when we apply the Wilcoxon test per dataset, comparing SC and GGC, we find that there is statistical difference in all the cases.

Once the algorithm has been applied to different datasets, to evaluate its performance, we have applied it to a real-world field, in this case, biomedical summarization.

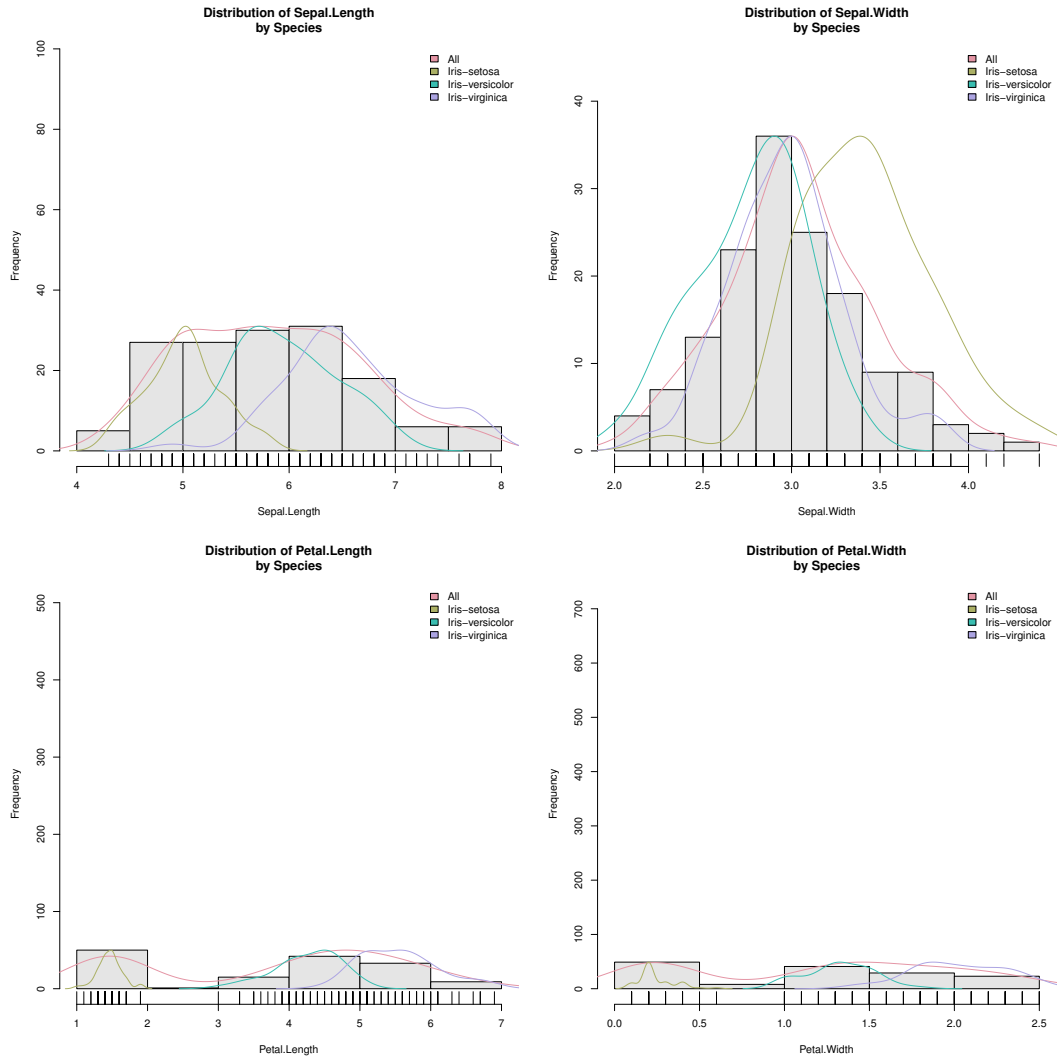


Figure 3.28: Histograms of the Iris Dataset. The three Gaussian functions that represent the classes distribution are also clearer separated for Patel-length and Petal-width variables. Versicolour and Virginica classes are too close together.

3.6 Applying Genetic Graph-based Clustering to Biomedical Summarization

The amount of biomedical literature available on the Internet, as in other disciplines, is growing exponentially [93]. Currently, the number of articles indexed in PubMed is over 19 millions. Biomedical experts experience information overloads and have difficulties in finding the information they need. In this situation, automatic summarization may be one of the techniques that could be used to alleviate this problem. Automatic summarization is the process of automatically generating an abbreviated, accurate representation of the content of a document (or set of documents). Given a set of documents that deal about the same topic (e.g., a disease, treatment and biological entity), summarization systems may extract the main concepts and provide a general perspective of the issue [213]. These summaries, which are not intended to

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit.
Ir	1000	2000	0.1	0.8-10^{-4}	50	0.99
Wn	100	20000	0.4	0.01-10^{-4}	50	1
Hw	20	20000	0.9	0.2-10^{-4}	5	0.90

Table 3.8: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in GGC algorithm for the different real datasets and the best fitness value obtained. The K value of the KNN-Minimal Cut fitness is always set to 2. The tournament size has also been fixed to 2.

	Iris	Wine	Handwriting
K-Means best classification	89.33%	95.50 %	50.83 %
EM best classification	**96.67**%	*97.19*%	35.43 %
Spectral Clustering best classification	89.33%	95.50%	*73.55*%
GGC best classification	▲*92*%	▲**100**%	▲**99**%

Table 3.9: Best accuracy values obtained by each algorithm during the experimental results applied to the UCI datasets. Statistical significant improvements are indicated by a ▲ symbol.

substitute the original document, may help researchers to quickly anticipate the content of the documents before deciding which of the documents to read further.

Summarization methods using conceptual representations have shown to outperform traditional text-level representations [162]. Representing the text as a set of concepts allows better capturing the meaning of the document. Moreover, these representations may be enriched with semantic relations between concepts (i.e., synonymy, hypernymy, co-occurrency and others) to build a domain-specific graph representation that accurately capture the meaning of the text to be summarized [162, 168, 178].

Automatic summarization of biomedical text makes use of domain-specific knowledge that is obtained from external sources (e.g., the Unified Medical Language System (UMLS)[4], the Systematized Nomenclature of Medicine - Clinical Terms (SNOMED-CT) [3]) or the Medical Subject Headings [1]) in order to model the text to be summarized [8, 182]. In this way, the documents are usually represented as graphs of biomedical concepts (the nodes in the graph) and relations among them (the links in the graph). A clustering method is then run over the graph to discover the *centroid* (i.e., the set of concepts that are most important to the cluster of documents) [161], and this is next use to extract the most relevant sentences from the original documents.

3.7 Summarization method

We use the summarization system presented in [162], which is briefly explained below and depicted in Figure 3.30. This system has been specially designed for summarization of biomedical literature. This new work modifies the clustering step of the original model using the GTC algorithm instead.

It consists of the following steps:

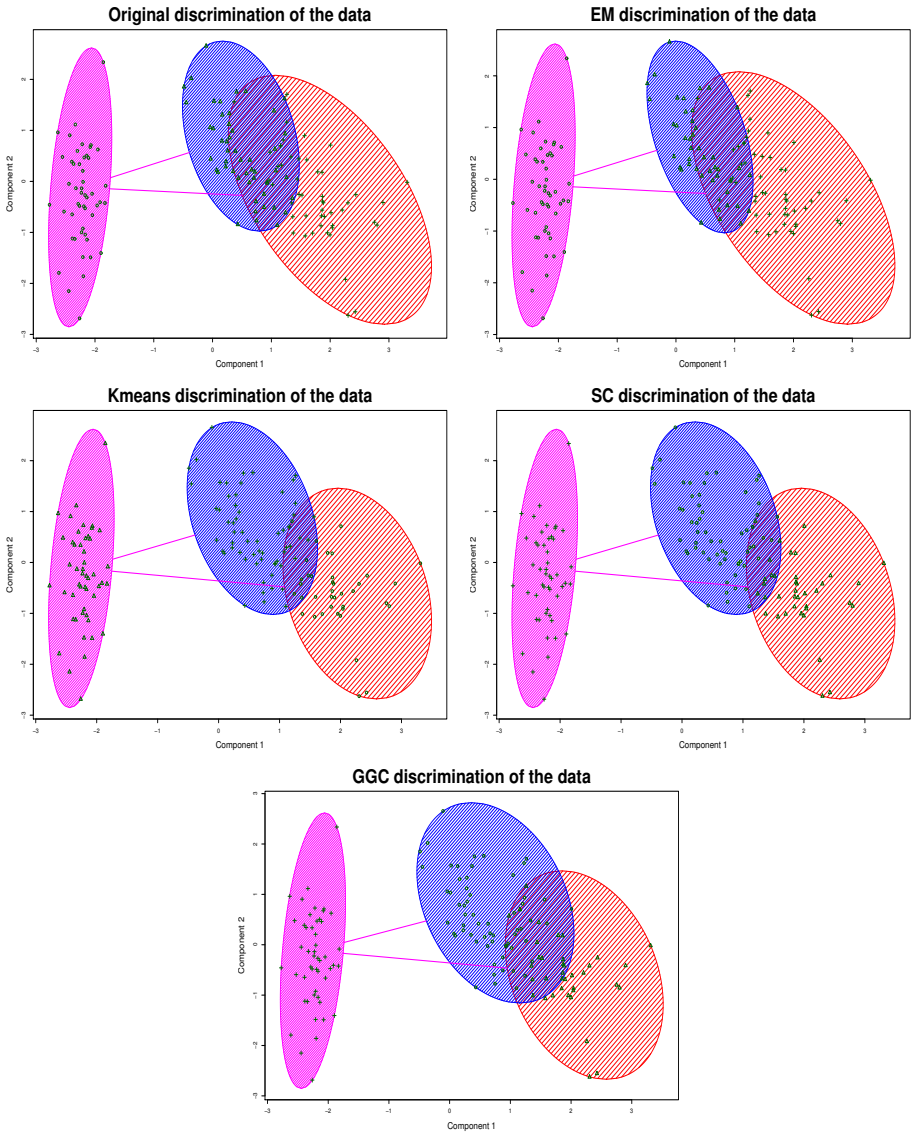


Figure 3.29: Discrimination of the data obtained by the different algorithms (from left to right and from top to bottom: The original Iris classification, the results from EM, the results from K-means, the results from Spectral Clustering and the results from the Genetic Algorithm) projected over 2 principal components.

1. **Document preprocessing:** In this step, irrelevant sections of the document (i.e., those that do not provide important information for the summary, such as *Competing Interests* or *Acknowledgements*) are removed. Abbreviations and acronyms are detected and expanded, and the *title*, *abstract*, and *body* sections are separated.

2. **Concept recognition:** The text in the document body is mapped to concepts from the UMLS Metathesaurus and semantic types from the UMLS Semantic Network [153], using MetaMap [2]. MetaMap is a software to discover UMLS Metathesaurus concepts which is used in text. MetaMap is invoked using the -y disambiguation option, which implements the Journal Descriptor Indexing methodology [92] and allows MetaMap to solve ambiguous mappings. UMLS concepts belonging to very general semantic types (e.g., *Spatial concept*

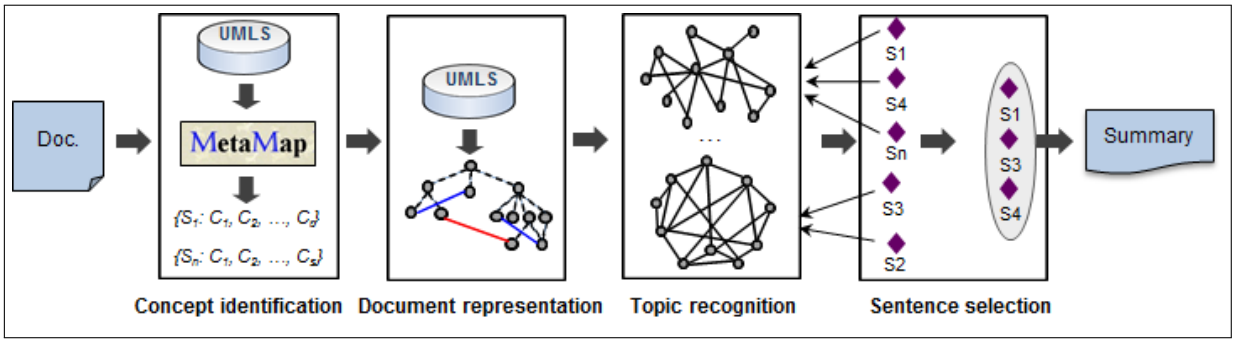


Figure 3.30: Architecture of the graph-based summarization system.

or *Language*) are ignored.

3. **Document representation:** For each sentence, each UMLS concept is extended with their hypernyms. All the hierarchies for each sentence are then merged to create *a sentence graph*, where the nodes represent domain concepts and the edges represent *is-a* relations between them. Next, the different sentence graphs are merged to build a single *document graph*. In this graph, new edges are added representing the following types of relations between UMLS concepts:

   - Relations between semantic types from the UMLS Semantic Network.
   - Relations between concepts from the UMLS Metathesaurus.

   Next, each edge is assigned a weight in [0,1], as shown in equation 3.6. The weight of an edge e representing an *is-a* relation between two vertices, N_i and N_j (where N_i is a parent of N_j), is calculated as the ratio of the depth of N_i to the depth of N_j from the root of their hierarchy. The weight of an edge representing any other relation (i.e., *associated with* and *related to*) between pairs of leaf vertices is always 1.

$$d(N_i, N_j) = \begin{cases} \frac{depth(N_i)}{depth(N_j)} & is\_a\ relation \\ 1 & otherwise \end{cases} \tag{3.6}$$

   To illustrate this process, Figure 3.31 shows the document graph for the following text from [91]:

   > *Interactions among LRF-1, JunB, c-Jun, and c-Fos define a regulatory program in the G1 phase of liver regeneration. In regenerating liver, a physiologically normal model of cell growth, LRF-1, JunB, c-Jun, and c-Fos among Jun/Fos/LRF-1 family members are induced posthepatectomy. In liver cells, high levels of c-Fos/c-Jun, c-Fos/JunB, LRF-1/c-Jun, and LRF-1/JunB complexes are present for several hours after the G0/G1 transition, and the relative level of LRF-1/JunB complexes increases during G1. We provide evidence for dramatic differences in promoter-specific activation by LRF-1- and c-Fos-containing complexes. LRF-1 in combination with either Jun protein strongly activates a cyclic AMP response element-containing promoter which c-Fos/Jun does not activate.*

4. **Clustering and topic recognition:** Once the graph has been generated, different clustering techniques can be applied to group the different concepts extracted from the text,

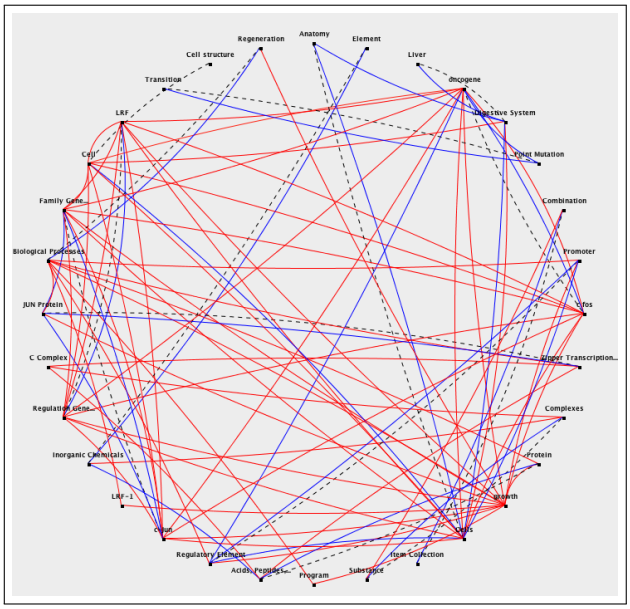


Figure 3.31: Example document graph. Dashed lines represent hypernymy relations; red lines represent Metathesaurus relations; and blue lines represent Semantic Network relations.

with the aim of identifying the different *topics* or *themes* that are dealt with in the text. In this work, a Genetic Text Clustering (GTC) and GGC algorithms have been tested (see Section 4).

Regardless of the clustering algorithm that is applied, the *salience* of each node in the graph may be calculated, using the Equation 3.7, as the sum of the weights of the edges that are connected to it. This salience is a measure of the node degree centrality.

$$Salience(N_i) = \sum_{j|N_j} d(N_i, N_j) \tag{3.7}$$

5. **Sentence selection:** The last step consists of computing the similarity between each sentence graph (S_i) and each cluster (C_j), and selecting the sentences for the summary based on these similarities. To compute sentence-to-cluster similarity, we add the salience of the common concepts between the sentence graph and the cluster. Finally, a single score for each sentence is calculated, as the sum of its similarity to each cluster adjusted to the cluster size (see Equation 3.8). The N sentences with highest scores are then selected for the summary.

$$Scr(S_j) = \sum_{C_i} \frac{sim(C_i, S_j)}{|C_i|} \tag{3.8}$$

3.8 Evaluation method

One of the most difficult and costly tasks in text summarization is to evaluate the automatically generated summaries. Deciding whether a summary has a good quality is very subjective, and

there is no agreement about the evaluation criteria that should be adopted [167]. Summarization evaluation techniques may be classified into two broad categories:

- **Intrinsic:** directly related to the quality of summarization.
- **Extrinsic:** concerned with the function or task in which the summaries are used, for instance, relevance assessment or reading comprehension.

This work is oriented to intrinsic summarization because the method used is not designed for any specific task. Intrinsic summarization techniques test the summarization focusing on two desirable properties of the summary [127]:

- **Coherence:** refers to text readability and cohesion.
- **Informativeness:** measures how much information from the source is preserved in the summary.

The evaluation of the summaries may be manual, however, this process requires human judges that need to be expert in the domain of the documents. Human evaluation requires to read both the summaries and the original documents to interpret the texts and extract the salient information, which is very time-consuming. It has also been proven difficult and highly subjective [99].

As a consequence, automatic metrics are usually employed to evaluate the quality of automatic summaries. However, these metrics only measure informativeness [188]. Research in automatic evaluation of coherence is still very preliminary [160].

In this work, the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) package [117] is used to evaluate the informativeness of the automatic summaries. ROUGE compares an automatic summary (called *peer*) with one or more human-made summaries (called *models* or *reference* summaries) and uses the proportion of n-grams in common between the peer and model summaries to estimate the content that is shared between them. The ROUGE metrics produce a value in [0,1], where higher values are preferred, as they indicate a greater content overlap between the peer and model summaries. The following ROUGE metrics are used in this work: ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4). R-N evaluates N-gram occurrence, where N stands for the length of the n-gram. Finally, R-SU4 evaluates "skip bigrams", that is, pairs of words having intervening word gaps no larger than four words.

3.8.1 Evaluation corpus

To evaluate the automatic summaries, we use a collection of 150 biomedical scientific articles randomly selected from the BioMed Central full-text corpus for text mining research [6]. This corpus contains approximately 85,000 papers of peer-reviewed biomedical research, available in XML structured format, which allowed us to easily identify the title, abstract, figures, tables, captions, citation references, abbreviations, competing interests and bibliography sections. As stated in [116], the document sample size is large enough to allow significant evaluation results.

Parameter	Experiment 1	Experiment 2
Breed	20	50
Crossover	0.9	0.8
Generation	1000	500
Mutation	0.2	0.15
Population	1000	900

Table 3.10: Values for both experiments of GGC algorithm.

As done in previous works [162, 168], the abstracts of the articles are used as gold standard (i.e., as model summaries for the ROUGE evaluation). Such abstracts, written by the authors of the articles, are supposed to summarize the main points of the documents.

3.9 Experiments with GGC

Two experiments have been carried out to compare different perspectives of the algorithms (see Table 3.10). The first experiment is focused on an intensive search during the clustering process (the population and the number of generations is high), while the second uses a relaxed search (the population and the number of generations is lower). Since the GGC algorithm needs the number of clusters (k), different experiments have been carried out with the number of clusters ranging from 2 to 9. This will allow us to empirically set the best value for k.

In order to evaluate the adequacy of our approach, the summaries generated by the summarizer have been compared to those produced by other summarization systems on the same evaluation collection. The first is a commercial application, *Microsoft AutoSummarize*, which uses a tradition term-frequency based approach. The rest are two baselines: *Lead* (which chooses the first sentences of the document to generate the summary) and *Random* (which chooses random sentences of the text to generate the summary).

3.9.1 Evaluation Results and Discussion

Tables 3.12 and 3.13 present two different experiments (whose genetic parameters are shown in Table 3.10) carried out for different values of k. These tables show the average results for each value of k, for the best k and for the other techniques. Each document has been processed 50 times per k value and the document with higher fitness value has been chosen for the evaluation phase (the average fitnesses per k and experiment are shown in Table 3.11).

Experiment 1 (see Table 3.12) shows good results compared with random baseline using the ROUGE-2 metric, however, the results are generally bad specially when the ROUGE-SU4 metric is applied. Choosing the best result for each value of k and document (see "Best K" in Table 3.12), the results, using the ROUGE-2 metric, are the best compared with the rest of algorithms. However, Lead baseline achieves better results according to ROUGE-SU4 metric.

Experiment 2 (see Table 3.13) shows better results than Experiment 1 for both metrics. According to ROUGE-2 metric, k from 2 to 8 have better results than the rest of algorithms. However, according to ROUGE-SU4 metric, Lead baseline is the best (compared with k from

	Average Fitness Ex1	Average Fitness Ex2
k=2	0.3706	0.3545
k=3	0.2939	0.2819
k=4	0.2429	0.2305
k=5	0.2059	0.1956
k=6	0.1786	0.1705
k=7	0.1584	0.1507
k=8	0.1431	0.1345
k=9	0.1308	0.1225

Table 3.11: Average best fitness values achieves by each value of k and the average values of the best results per k.

	ROUGE-2	ROUGE-SU4
k=2	0.211	0.175
k=3	0.215	0.175
k=4	0.223	0.182
k=5	0.219	0.176
k=6	0.216	0.177
k=7	0.213	0.173
k=8	0.208	0.170
k=9	0.212	0.175
Best k	**0.300**	*0.250*
LEAD	*0.257*	**0.265**
AutoSummarize	0.245	0.232
Random baseline	0.173	0.230

Table 3.12: Experiment 1. Results from the application of GGC algorithm for different values of k and the best value obtained. These results are compared with a commercial application (Microsoft AutoSummarize) and two baselines (Lead and Random). The best scores are shown in bold and the second best results in italics.

2 to 8), although, in this experiment, the rest of the algorithms are beaten. Choosing again the best result for each value of k (see "Best K" in Table 3.13) the algorithm achieves the best scores in both metrics.

As Table 3.11 shows the fitness values of the first experiment are generally higher than the values of the second experiment. Comparing these results with Tables 3.12 and 3.13, an over-fitting problem generated by the algorithm might be the cause of the high different of values between the two algorithms. Over-fitting is a classical problem in Data Mining process [111] which is usually avoided by using methods such as cross-validation [76] or adjusting the parameters of the algorithm. These results show that it is not necessary a deep search because it might produce an undesirable over-fitting problem.

3.9.2 The election of k

As Table 3.12 and 3.13 show, the different values of k have similar results. Moreover, we have observed that the best k value strongly varies across different documents, so that there is not a

	ROUGE-2	ROUGE-SU4
k=2	0.261	0.244
k=3	0.268	0.254
k=4	*0.273*	0.255
k=5	0.270	0.252
k=6	0.264	0.248
k=7	0.266	0.250
k=8	0.264	0.245
k=9	0.252	0.238
Best k	**0.346**	**0.319**
LEAD	0.257	*0.265*
AutoSummarize	0.245	0.232
Random baseline	0.173	0.230

Table 3.13: Experiment 2. Results from the application of GGC algorithm for different values of k and the best value obtained. These results are compared with a commercial application (Microsoft AutoSummarize) and two baselines (Lead and Random). The best scores are shown in bold and the second best results in italics.

	Ex1 Min	Ex1 Max	Ex1 SD	Ex2 Min	Ex2 Max	Ex2 SD
k=2	0.0588	0.9412	0.1878	0.0735	0.9265	0.1910
k=3	0.0240	0.9027	0.1800	0.0303	0.8774	0.1736
k=4	0.0017	0.8958	0.1805	0.0017	0.8876	0.1776
k=5	0.0013	0.8406	0.1721	0.0013	0.9167	0.1731
k=6	0.0011	0.8976	0.1587	0.0011	0.9271	0.1611
k=7	0.0010	0.8434	0.1523	0.0010	0.9235	0.1519
k=8	0.0008	0.8125	0.1401	0.0008	0.8406	0.1381
k=9	0.0007	0.7959	0.1327	0.0007	0.7917	0.1326

Table 3.14: Cluster statistics from Experiments 1 and 2. All values show percentages of cluster membership.

best k value for all documents.

On average, $k = 4$ produces the best summarization results. Since k is the number of clusters, the fitness function is not a good choice to decide the best value of k, because the fitness decreases when the number of clusters is increased (see Table 3.11). Some statistics have been extracted from the different solutions generated by the algorithm (see Table 3.14). These statistics give information about maximum, minimum and standard derivation percentages of cluster memberships (the average in these cases is always $1/k$). These results show that Experiment 2 produces more balanced clusters than Experiment 1 (see SD in Table 3.14), however, the best k value ($k = 4$), has in both cases a higher standard derivation which suggests that the clusters size can be highly variable. This has sense because there are concepts about some topics that do not belong to the main topic of the document. These statistics do not give any indication about the selection of the number of clusters before the ROUGE evaluation, however, the decision of this value is out of the scope of this work.

3.9.3 Discussion and Improvements

This work has combined a Genetic Graph-based Clustering (GGC) algorithm and a graph-based summarization process. This combination has been evaluated through two experiments: the first is a deep search of the genetic clustering algorithm and the second a relaxed search. The following conclusions have been extracted from this work:

- The new process also obtains better results than classical and commercial algorithms.
- A deep search during the clustering process causes over-fitting in the summarization process and affects negatively to the global results.
- A good selection of the original number of clusters causes a high improvement in the results.

There are also some issues which might be studied in the future:

- It is necessary to find a method to choose the number of clusters according the document features.
- The fitness function should add some other metrics related to other properties of the graph such as the salience.
- Finally, other summarization processes might be compared with the current methodology.

3.10 Experiments with SVM-Rank

Our previous work was focused on the comparison of different perspectives of the GGC algorithm in Biomedical Summarization [143]. Two experiments were carried out in order to compare whether a deep or a relaxed search was necessary to find good solutions (see Table 3.10 for the parameter settings). Due to GGC needs an initial number of clusters, different values of k

were compared during the experiments. This work has used the information provided by these experiments to generate a new model which automatically chooses the k value. To generate this model, we have used a learning-to-rank approach because we are interesting to detect the best k (ranking all the possible values). The model chosen is the SVM-Rank model. It has been trained with 100 documents. The training feature set is composed by: the number of clusters, the Genetic Algorithm parameters (Table 3.10), the number of nodes and the density of the graph. The rest of documents, i. e. 50 documents, have been used to test the model and compare it with other summarization systems.

The metric applied to the SVM-Rank model is the Radial Basis Function (RBF), defined as follows:

$$D_{RBF}(X, Y) = e^{\sigma\sqrt{\sum_{i=1}^{N}(X_i - Y_i)^2}} \tag{3.9}$$

Where X and Y are points, N is the number of coordinates, and σ is a normalization parameter. In this work, σ is set to 1.

Using this information we have compared the results of the original experiments, applied to the test set, against the new approach. In order to evaluate the adequacy of our approach, the summaries generated by the summarizer have been also compared to those produced by other summarization systems on the same evaluation collection. The first is a commercial application, *Microsoft AutoSummarize*, which uses a traditional term-frequency based approach. The rest are two baselines: *Lead* (which chooses the first sentences of the document to generate the summary) and *Random* (which chooses random sentences of the text to generate the summary).

3.10.1 Results and Discussion

Tables 3.15 and 3.16 present the two different experiments (whose genetic parameters have been shown in Table 3.10) carried out for different values of k and for the SVM-Rank model. The first experiment is the deep search (the algorithm parameters such as "Number of Generations" or "Population" are higher) while the second is the relaxed search (the parameters are lower). These tables show the average results for each value of k, for the new model (SVM-Rank), for the best k and for the other techniques. The clustering analysis has been carried out 50 times per document and k value. The solution with higher fitness value has been chosen from the evaluation phase.

Experiment 1 (see Table 3.15) shows good results compared with random baseline using the ROUGE-2 metric, however, the results are generally bad specially when the ROUGE-SU4 metric is applied. Choosing the best result for each value of k and document (see "Best k" in Table 3.15), the results, using the ROUGE-2 metric, are the best compared with the rest of algorithms. Using the new model, which automatically chooses the value of k (see "SVM-Rank k") the algorithm is the fourth of the ranking, behind to AutoSummarize for the ROUGE-2 metrics, i.e. the summarize bi-grams coincide with the abstract. Lead baseline achieves better results according to ROUGE-SU4 metric, i.e. the skip bi-grams generated by this algorithm are more similar to the abstract. This experiment has carried out a deep search in the solution space. The results achieved are generally bad compared with the rest of the algorithms. It is usual when a deep search is applied to genetic algorithm, because the algorithm falls on local minimums of the search space producing an over-fitting effect in the algorithm solutions.

Experiment 2 (see Table 3.16) shows better results than Experiment 1 for both metrics.

	ROUGE-2	ROUGE-SU4
k=2	0.221	0.176
k=3	0.230	0.183
k=4	0.225	0.182
k=5	0.229	0.179
k=6	0.221	0.177
k=7	0.219	0.173
k=8	0.209	0.164
k=9	0.214	0.172
SVM-Rank k	0.237	0.190
Best k	**0.303**	**0.248**
LEAD	**0.257**	**0.265**
AutoSummarize	*0.245*	*0.232*
Random baseline	0.173	0.230

Table 3.15: Experiment 1. Results from the application of GGC algorithm for different values of k and the best value obtained. These results are compared against a commercial application (Microsoft AutoSummarize) and two baselines (Lead and Random). The best scores are shown in bold and the third best results in italics.

According to ROUGE-2 metric, k from 2 to 8 have better results than the rest of algorithms. The new model obtains the best results (behind "Best k") compared with the rest of the algorithms. According to ROUGE-SU4 metric, the new model obtains the best results, followed by Lead baseline. In this experiment, the rest of the algorithms are beaten. Choosing again the best result for each value of k (see "Best K" in Table 3.16) the algorithm achieves the best scores in both metrics.

Concluding, these experiments show that a relaxed search in the search space provides better general results. Adding the SVM-Rank model, we are also able to improve the original results providing better summaries according to ROUGE metrics. Finally, a deep search produces over-fitting in the set of solutions, generating worse summaries according to ROUGE.

3.10.2 Discussion

This work has combined a Genetic Graph-based Clustering (GGC) algorithm, a graph-based summarization process and a learning-to-rank method. This combination has been evaluated through two experiments: the first is a deep search of the genetic clustering algorithm and the second a relaxed search. The following conclusions have been extracted from this work:

- The learning-to-rank technique helps to improve the original results. Combined with a relaxed search, it beats classical and commercial algorithms according to ROUGE-2 and ROUGE-SU4 metrics.

- A deep search during the clustering process causes over-fitting in the summarization process and affects negatively to the global results.

	ROUGE-2	ROUGE-SU4
k=2	0.268	0.251
k=3	0.268	0.249
k=4	0.271	0.249
k=5	*0.281*	0.261
k=6	0.262	0.243
k=7	0.277	0.255
k=8	0.271	0.246
k=9	0.262	0.244
SVM-Rank k	**0.284**	**0.266**
Best k	**0.358**	**0.328**
LEAD	0.257	*0.265*
AutoSummarize	0.245	0.232
Random baseline	0.173	0.230

Table 3.16: Experiment 2. Results from the application of GGC algorithm for different values of k and the best value obtained. These results are compared against a commercial application (Microsoft AutoSummarize) and two baselines (Lead and Random). The best scores are shown in bold and the third best results in italics.

3.11 The Genetic Text Clustering (GTC) algorithm

Once the document graph is generated (see step 3 of the summarization method), a clustering algorithm is applied to separate the topics within it. This is done by grouping together in the same topic those concepts that are highly interconnected. The new algorithm combines the degree centrality of the nodes in the graph (as measured by their salience, see Equation 3.7) and the graph continuity in order to extract the main topics and keep the continuity among them. This algorithm is applied in three steps:

1. **Similarity Graph generation:** a similarity function (usually based on a kernel) is applied to the data instances (i.e., the domain concepts), connecting all the points with each other. It generates the Similarity Graph.

2. **Genetic search:** Giving an initial number of clusters $k_{clusters}$, the GA generates an initial population of possible solutions and evolves them using a fitness function to guide the algorithm to find the best solution. It stops when a good solution is found, or a maximum number of generations is reached.

3. **Clustering association:** The solution with the highest fitness value is chosen as a solution of the algorithm and the data instances are assigned to the $k_{clusters}$ clusters according to the solution chosen.

3.11.1 Encoding and Genetic operators

The Encoding is a simple label-based representation [90]. Each individual is a n-dimensional vector (where n is the number of data instances) which has integer values between 1 and the

number of clusters. They represent a possible solution. i.e., a cluster selection for each data instance of the dataset.

During the evolution process, the operators can create invalid individuals. These individuals represent solutions where one or more clusters have no elements. In this problem of partitional clustering, these solutions are not valid because the number of clusters is initially given. In this work, no attempt to repair invalid solutions is done. Instead, to avoid the invalid individuals generation problem, they receive a 0 fitness value. The operators used can be briefly summarized as follows:

- **Selection**: The selection process selects a subset of the best individuals. These chromosomes are reproduced and also passed to the next generation. It is called a $(\mu+\lambda)$ selection [45], where μ represents those chromosomes which are chosen, and λ the new chromosomes generated.
- **Crossover**: The crossover operation exchanges strings of numbers between the two chromosomes (both strings have the same length). To reduce the search space, it previously *relabels* those individuals which have different numerical values but represent the same solution (i.e. if there are two chromosomes which represent the same solution but the labels of their clusters are different, these labels are changed in order to maximize the similarity between them).
- **Mutation**: The mutation randomly chooses different chromosomes to change the values of some of their alleles. The new value is a random number between 1 and the number of clusters.

3.11.2 The Fitness Function

The fitness function is an hybrid fitness divided in two parts: i) improving the data continuity degree and ii) improving the total salience of the clusters that are generated.

The continuity is guaranteed through a KNN (K-Nearest Neighbour) metric. To control the clusters salience, this metric has been added to the fitness (see Equation 3.11). It guarantees that the clusters are composed of concepts which are relevant in the graph. The K value for KNN is initially given by the user, nevertheless, in this work we have fixed it to 2 because it is the minimal value to guarantee the continuity and additionally it avoids over-fitting. The Genetic Algorithm maximizes the value of:

$$\frac{TotalKNN \cdot TotalSal}{|C|} \tag{3.10}$$

where:

$$TotalSal = \sum_{C_\alpha \in C} \frac{\sum_{N_i \in C_\alpha} Sal(N_i)}{max_{C_\alpha}\{\sum_{N_i \in C_\alpha} Sal(N_i)\}} \tag{3.11}$$

$$TotalKNN = \sum_{x \in C} \frac{|\{y | y \in \Gamma(x) \wedge y \in C_x\}|}{|\Gamma(x)|} \tag{3.12}$$

In these formulas, $Sal(N_i)$ represents the salience of the node i (see Equation 3.7), C represents the set of clusters, C_α represents a cluster and $\Gamma(x)$ represents the neighborhood of the element x.

Parameter	GGC & GTC
Breed	50
Crossover Prob.	0.8
Generation	500
Mutation Prob.	0.15
Population	900

Table 3.17: Parameter values for both GGC and GTC algorithms.

3.12 Experiments wit GTC

The previous work was focused on comparing different perspectives of the GGC application [143]. Two experiments where carried out in order to compare whether a deep or a relaxed search was necessary to find good solutions (see Table 3.10 for the parameter settings). Due to that GGC needs an initial number of clusters, different values of k were compared during the experiments. This work has used the information provided by these experiments and has compared both methods using the same relaxed search for both algorithms (see Table 3.17). Using this information, we compared the results of both algorithms applied to 150 biomedical documents. In order to evaluate the adequacy of our approach, the summaries generated by the summarizer have been also compared to those produced by other summarization systems on the same evaluation collection. The first is a commercial application, *Microsoft AutoSummarize*, which uses a tradition term-frequency based approach. The second is a research application, LexRank [62]. The rest are two baselines: *Lead* (which chooses the first sentences of the document to generate the summary) and *Random* (which chooses random sentences of the text to generate the summary).

3.12.1 Results and Discussion

Table 3.17 shows the Genetic Parameter selection for both algorithms, these parameters have been selected in order to carry out a relaxed search. Table 3.18 presents the results for each algorithm separated by k-values. This table shows the average results for each value of k, for the new model, for the best k and for the other techniques. The best k solution represents the best solution per document comparing the different solutions per k value. The clustering analysis has been carried out 50 times per document and k value. The solution with higher fitness value has been chosen for the evaluation phase.

GGC (see Table 3.18) shows good results compared with all baselines and systems according to ROUGE-2 metric; however, the results are generally worse when the ROUGE-SU4 metric is applied. LexRank obtains better results than GGC (for k from 2 to 9). Choosing the best result for each value of k and document (see "Best k" in Table 3.18), the results, using both metrics, are the best compared with the rest of algorithms.

GTC (see Table 3.18) shows better results than GGC for both metrics. According to ROUGE-2 metric, the "Best k" value outperforms the results of the GGC algorithm. Also, according to ROUGE-SU4 metric, the value of the different k solutions is closer to the Lead baseline and LexRank values, which are the best (compared with k from 2 to 9), although, in

	ROUGE-2	ROUGE-SU4
GGC k=2	0.261	0.244
GGC k=3	0.269	0.254
GGC k=4	0.273	0.255
GGC k=5	0.270	0.252
GGC k=6	0.264	0.248
GGC k=7	0.267	0.250
GGC k=8	0.264	0.245
GGC k=9	0.253	0.238
GGC Best k	**0.347**	**0.319**
GTC k=2	0.278	0.260
GTC k=3	0.260	0.245
GTC k=4	0.271	0.253
GTC k=5	0.269	0.254
GTC k=6	0.270	0.254
GTC k=7	0.278	0.261
GTC k=8	0.262	0.249
GTC k=9	0.274	0.254
GTC Best k	**0.357**	**0.329**
LexRank	*0.308*	*0.277*
LEAD	0.257	0.265
AutoSummarize	0.245	0.232
Random baseline	0.173	0.230

Table 3.18: Results from the application of GGC and GTC algorithms for different values of k and the best value obtained. These results are compared with a commercial application (Microsoft AutoSummarize), a research prototype (LexRank), and two baselines (Lead and Random). The best scores are shown in bold and the second best results in italics.

this experiment, the rest of the algorithms are beaten. Choosing again the best result for each value of k (see "Best k" in Table 3.18) the algorithm achieves the best scores in both metrics.

The k value is not always the same for both algorithms (GGC and GTC). This should be a consequence of the algorithm objectives. GGC is focused only on the cluster continuity while GTC is also focused on the centrality degree, thus generating different solutions.

These results show that GGC, which is totally focused on the cluster continuity, obtains good results; however, if the clustering is also focused on the total salience of the concepts, the results improve. GTC concentrates on the salience of the graph, joining concepts which are relevant within the graph. This improves the number of important concepts in the chosen sentences giving more information to the reader about the text; while GGC is focused only on the continuity of the concepts, joining these concepts which are related with each other. These results show that both approaches are mutually beneficial. Other important consideration of these results is the parameter selection. The parameters have been chosen for a relaxed search, i.e., they look for good solutions in the search space using a low number of individuals and population. This also concerns to the algorithm speed, making the algorithm to find a good solution faster than using a deep search.

3.12.2 Discussion

This work has performs a Genetic Graph-based Clustering (GGC) algorithm, using the Graph Salience to increase the topic relevance during the clustering process. The new algorithm, called Genetic Text Clustering (GTC) has shown that the combination of continuity-based measures and degree centrality (salience) obtains better results than the original techniques. The new graph-based summarization process has been evaluated using 150 biomedical documents which has shown that the algorithm also obtains better results than other research and commercial techniques. The following main conclusions have been extracted:

- The topic importance is highly relevance during the summarization process. The combination with continuity-based clustering helps to determine the importance of the sentences in the summary by providing more information about the relevance of the different topics that are dealt with in the text.
- This new methodology beats classical and commercial algorithms.

Chapter 4

Multi-Objective Genetic Graph-based Approaches

"Erfahrung ohne Theorie ist blind,
aber Theorie ohne Erfahrung ist bloßes intellektuelles Spiel."

- Immanuel Kant

In previous chapter, we have proposed a Genetic Graph-based Clustering algorithm (GGC) [140] to deal with the robustness problem that appears in SC algorithm. It combines the classical K-Nearest Neighbourhood (KNN) algorithm [111] and the Minimal Cut measure [177] to search the best cut of the graph.

GGC uses the same graph representation that SC and also improves the robustness of the clustering results related to the metric used to measure the data similarity. However, this algorithm has the same memory usage problems than SC: it generates a matrix comparing all the data instances pair to pair, whether the problem is focused on large datasets, this matrix becomes extremely big and it is difficult to store (and therefore to compute) all its information.

In this chapter we propose a new algorithm named Multi-Objective Genetic Graph-based Clustering Algorithm (MOGGC). It is based on GGC and combines Multi-Objective Genetic Algorithms (MOGA) [42] with graph-continuity metrics to achieve two goals: to reduce the memory consumption and to mantain solution quality in comparison to GGC. In order to assess MOGGC performance, we have compared it against three classical clustering algorithms (K-means, EM and SC) and the original GGC. The experimentation reported in this chapter involves synthetic and well-known UCI datasets.

4.1 The Multi-Objective Genetic Graph-based Clustering (MOGGC) algorithm

This section describes the Multi-Objective Genetic Graph-based Clustering (MOGGC) algorithm. MOGGC uses the SPEA2 [212] algorithm for the genetic evolution of the set of solutions which are encoded as the population. This algorithm is a MOGA which improves the results of the convergence through the Pareto Front.

SPEA2 [212] starts with two populations P_0 and $\overline{P_0}$, the first is known as the internal population and the second is the external population which is initially empty (see line 1 of Algorithm 7). During each generation, the algorithm calculates the fitness of both populations (P_t and $\overline{P_t}$), and takes the non-dominant individuals to the external population of the next generation (see lines 3 and 4 of Algorithm 7). If the external population is bigger than the initial size it is reduced and when the size is smaller it is filled with dominated individuals of the original populations using a truncation method (see lines 5 to 9 of Algorithm 7). Next, it fills a mating pool with individuals of $\overline{P_{t+1}}$ selected by binary tournament and applies the genetic operations to generate the new population P_{t+1} (see lines 13 and 14 of Algorithm 7). This algorithm keeps a copy of the best Pareto Front selection of each generation in the external population.

As K-means requires, it is necessary to give an initial number of clusters to MOGGC. It begins with a K_{size}-Similarity Graph (see Section 2.4.3) in the same way that the Spectral Clustering algorithm makes. The population is a set of possible solutions (partitions) which evolves until the best solution is achieved, or the maximum number of generations is reached. The fitness function is a quality measure for those solutions.

MOGGC algorithm is applied in three steps:

1. **Similarity Graph generation:** a Similarity Function (usually based on a kernel) is applied to the data instances (i.e., the domain concepts), connecting all the points with each other. It generates the Similarity Graph.
2. **Genetic search:** Giving an initial number of clusters $k_{clusters}$, the GA generates an initial population of possible solutions and evolves them using a fitness function to guide the algorithm to find the best solution. It stops when a good solution is found, or a maximum number of generations is reached.
3. **Clustering association:** The solution with the highest fitness value is chosen as a solution of the algorithm and the data instance are assigned to the $k_{clusters}$ clusters according to the solution chosen. In this case, the solution is chosen in the Pareto Front.

4.1.1 Encoding and Genetic operators

The encoding is a simple label-based representation [90]. Each individual is a n-dimensional vector (where n is the number of data instances) which has integer values between 1 and the number of clusters. They represent a cluster selection for the dataset.

During the evolution process, the operators can create invalid individuals, similar to GGC. They also receive a 0 fitness value. The operators used can be briefly summarized as follows:

- **Selection**: The selection process is a tournament selection.
- **Crossover**: The crossover exchanges strings of numbers between the two chromosomes (both strings have the same length).
- **Mutation**: The mutation is adaptive. It works as follows:

Algorithm 7 Pseudo-code of the SPEA2 algorithm [212]

Input: N (population set); $\overline{N}$ (archive size); T (generations)
Output: A (non-dominated set) .

```
P_0 = random population; \overline{P}_0 = ∅;
for t = 0 → T do
    Calculate Fitness of P_t and \overline{P_t}.
    Copy non-dominated individuals in P_t and \overline{P_t} to \overline{P_{t+1}}
    if size(\overline{P_{t+1}}) > \overline{N} then
        reduce \overline{P_{t+1}}
    else
        Fill \overline{P_{t+1}} with dominated individuals in P_t and \overline{P_t}
    end if
    if t == T or any stopping condition is satisfied then
        Break the loop.
    end if
    Fill the mating pool with individuals of \overline{P_{t+1}} selected by binary tournament.
    Apply the recombination and mutation to the mating pool and set P_{t+1} to the resulting
    population.
end for
return A = {non-dominated individuals in P_{t+1}}
```

1. For each chromosome, it randomly chooses if the mutation is applied. The mutation probability is fixed at the beginning.
2. When a chromosome is chosen, it decides the alleles which are mutated. The decision considers the probability of the allele to belong to the cluster which have assigned. If the probability is high, the allele has a low probability of mutate and vice versa. In this algorithm, this probability is calculated applying the metric defined in the fitness function to one allele.
3. The alleles are mutated. The new value is a random number between 1 and the number of clusters.

4.1.2 The Fitness Objectives

The fitness function is divided in two objectives: improve the data continuity degree and improve the cluster separation. It uses a K_{size}-Similarity Graph [193] as a starting point like other Spectral Clustering techniques. The K_{size} value limits the memory used to a matrix $K_{size} \times N$ where N is the number of data instances.

4.1.2.1 Data Continuity Degree

This objective function is applied to each cluster. It calculates the total edges sum for each minimal spanning tree of each connected component of the K_{size}-Graph G (see Algorithm 8). Starting in the first node (it supposes, without loss of generality, that the nodes are numerically ordered), the algorithm generates two lists: the first initially contains all nodes and the second

Algorithm 8 Data Continuity Degree Algorithm

Input: C cluster with an order relationship
Output: ν (connectivity factor) .

```
1: Let L1 = C and L2 = ∅ and set ν = 1;
2: Move the first element of L1 to L2;
3: while L1 ≠ ∅ or L2 ≠ ∅ do
4:     Set v_i = the first element of L2 (Extract it from the list);
5:     for v_j ∈ G do
6:         if vj ∈ L1 and v_j > v_i then
7:             Move v_j from L1 to L2;
8:             ν + +;
9:         end if
10:    end for
11:    if L2 = ∅ then
12:        if L1 = ∅ then
13:            break;
14:        end if
15:        Move the first element of L1 to L2;
16:    end if
17: end while
18: return ν/|C|;
```

is empty (see line 1 of Algorithm 8). While any of the lists contains at least one element, the first list will give to the second all nodes connected within the neighbourhood of the current node and internally will count the minimal spanning tree edges (see lines 3 to 9 of Algorithm 8). Due to the graph is not full-connected, this process will follow with each connected component (see lines 10 to 17 of Algorithm 8). This metric measures the continuity of the data as a graph structure inside the clusters. The arithmetic average value of the metric is the result of this objective.

4.1.2.2 Clusters Separation

The second objective of the fitness function is to maximize the cluster separation. To ensure it, the following metric has been applied to each cluster:

$$\frac{\sum_{v_i \in C} \frac{\sum_{v_j \in G}\{w_{ij}|\ v_j \notin C\}}{|G|-|C|}}{|C|} \tag{4.1}$$

where C is a cluster, G is the K_{size}-Graph, v_i is the vertex i, w_{ij} is the edge weight value from i to j. It calculates the arithmetic average value of the edge weights between the different clusters.

The MOGA implementation is necessary due to our current objectives from previous fitness functions. The two objectives are opposites: the first tries to improve the inter-clusters distance and the second the intra-cluster distance. In the first case, a single cluster would guarantee a maximum value while, in the second case, a cluster per instance would guarantee the maximum value.

4.1.3 Differences between MOGGC and GGC algorithm

The most important differences between MOGGC and GGC algorithm are based on structural differences of the algorithms and the fitness functions.

On the one hand, the structure of MOGGC is a MOGA while the structure of GGC is a simple GA. However, the encoding and the operations are the same for both algorithms.

On the other hand, the fitness functions are highly different. While MOGGC uses a K_{size}-Similarity Graph, GGC uses a full-Similarity Graph (see Section 2.4.3). The main differences between the two graphs is their memory size: a full-Similarity Graph is an N^2 matrix where N is the number of instances while a K_{size}-Similarity Graph is a $K_{size} \times N$ matrix where $K_{size} << N$. In the first case, the Similarity Graph grows exponentially while, in the second case, it grows linearly.

The fitness calculus are also different, MOGGC uses the Data Continuity Degree and the Clustering Separation metrics and GGC uses the Minimal Cut metric and a K-Nearest Neighbour (KNN) approximation [140] to calculate a single fitness value which is an equilibrium of these two measures.

4.2 Experimental Results

This section shows the experimental results. The first part presents the datasets which have been used to test the algorithm. The second shows the Pareto Front study. The third describes the evaluation metrics and the experimental set-up. The fourth part shows the results on the synthetic and real-world datasets which have been taken from the literature. Finally, the last part shows a comparison between the memory cost of GGC and MOGGC.

4.2.1 Evaluation Datasets

This section describes the different datasets which have been used for the algorithm testing phase. Synthetic and Real World datasets have been used to check the algorithm accuracy. These datasets have been extracted from different works related to clustering problems, as was explained in Chapter 3.

4.2.1.1 Synthetic datasets

The datasets which have been chosen are (see Figure 4.2):

- *Aggregation* (Ag) [78]: This dataset is composed by 7 clusters, some of them can be separated by parametric clustering.
- *Jain* (Jn) [96]: This dataset is composed by two surfaces with different density and a clear separation.

- *R15* [192]: This dataset is divided in 15 clusters which are clearer separated.
- *Spiral* (Sp) [37]: In this case, there are 3 spirals close to each other.

4.2.1.2 Real-World datasets

The datasets which have been chosen have been extracted from the UCI database [70]. They are the following:

- *Iris* (Ir): Contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Each instance has 4 attributes. This well known dataset has been used in several clustering works [115].
- *Wine* (Wn): Contains 3 classes with 13 attributes each and 178 instances. It also has been used in some clustering works as [29].
- *Glass* (Gl): Contains 6 classes with 9 attributes each and 214 instances. It also has been analysed in some clustering works as [64]
- *Libras Movement* (LM): Contains 15 classes with 90 attributes each and 24 instances per class (total 360). It is identified for classification and clustering in the UCI database [70].
- *Ozone Level Detection* (OL): Contains 2 classes with 73 attributes and 2536 instances. It has been chosen because of its simplicity according to the number of classes.
- *Wine Quality* (WQ) [47]: Contains 6 classes with 11 attributes each and 4898 instances of white wine. it is also identified for classification and clustering in the UCI database [70].
- *Page Block* (PB): Contains 5 classes with 10 attributes each and 5473 instances. It has been chosen because of its complexity.

4.2.2 Choosing the solution from the Pareto Front

Due the necessity to choose one of the solutions from the Pareto Front, the experimental results show that the solution with the highest value of the Cluster Separation metric in the Pareto Front always obtains better accuracy values. Therefore, this value has been chosen as the algorithm solution.

In order to study the Pareto Front solution selection, we have repeated the original experiments relaxing the parameters of the genetic algorithms to generate a clearer Pareto Front (i.e. setting the parameters for a shallow search instead of a deep search). The algorithms have been carried out 100 times. These parameters achieve a relax solution which is worse than the final solutions but helps to study the Pareto Front. Per each dataset, we have selected five points (or levels) in the Pareto Front, which are: 1st and 5th the highest values of each objective; 3rd is the balanced value; 2nd and 4th are values between the maximum of each objective and the balanced value. See Figure 4.1 left, to see an example of the level selection for the Aggregation dataset. Level 1 is represented by the square symbol and the levels continue until the level 5 represented by the diamond symbol. Figure 4.1 right shows the accuracy results for each Level

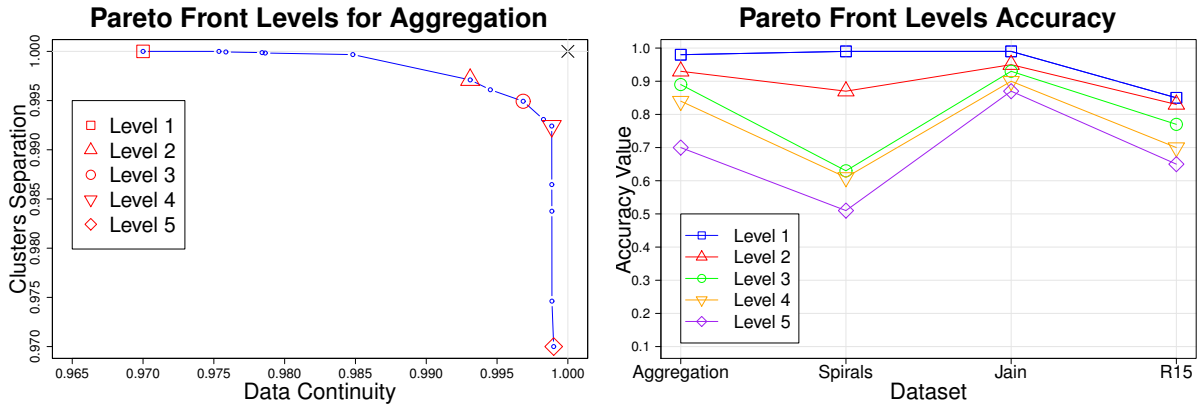


Figure 4.1: Pareto Front study: The left image shows an example of the levels chosen in the Pareto Front of Aggregation (square represents Level 1, triangle Level 2 and it continues until the diamond which represents Level 5). The right image shows the results of the 5 levels applied to the four synthetic datasets.

applied to the synthetic datasets. As we can see, Level 1 achieves the best Pareto Front results for all the datasets which explain the decision of choosing the solution which optimizes the Clustering Separation objective. Aggregation, Jain and R15 has close values for the different levels, however, Spirals dataset has bigger gaps between the values. It might be because this dataset has several local minimum in the solution space. R15 has lower accuracy than the rest, it is because this dataset has a bigger search space and the relax parameters are not closed to the solution.

4.2.3 Evaluation Techniques and Experimental Setup

The MOGGC algorithm has been compared against different clustering algorithms. These algorithms have been taken from the literature [125, 56, 155] and from our previous work [139]. The classical algorithms which have been chosen are: K-means, Expectation Maximization and Spectral Clustering. Also, it has been compared against the previous implementation of GGC algorithm [140].

The similarity between the clusters have been calculated using the following similarity metric:

$$sim(C_i, C_j) = \frac{1}{2}\left(\frac{\sum_{q=1}^{n} \delta_{C_i}^q \delta_{C_j}^q}{|C_i|} + \frac{\sum_{q=1}^{n} \delta_{C_i}^q \delta_{C_j}^q}{|C_j|}\right) \tag{4.2}$$

where n is the number of elements, C_i, C_j are the clusters which are compared, $|C_i|$ is the number of elements of cluster C_i, and $\delta_{C_i}^q$ is the Kronecker δ defined by:

$$\delta_{C_i}^q \equiv \delta_{C_i}(x_q) = \begin{cases} 0 & \text{if } x_q \notin C_i \\ 1 & \text{if } x_q \in C_i \end{cases}$$

where x_q is an element. The evaluation process calculates the maximum accuracy for all the algorithms. All of them have been executed 150 times per dataset. The metric which has been

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit.
Ag	100	2000	0.4	0.01-10^{-4}	50	0.99
Jn	100	500	0.4	0.2-10^{-4}	50	1.0
R15	200	2000	0.5	0.3-10^{-4}	50	0.98
Sp	100	500	0.4	0.01-10^{-4}	50	1.0

Table 4.1: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in GGC algorithm for the different real datasets and the best fitness value obtained. The tournament size is 2.

used for K-means and EM is the Euclidean Metric defined by:

$$||x_i - x_j|| = \sqrt{\sum_{q=1}^{d}(x_i^q - x_j^q)^2} \tag{4.3}$$

Where $x_i = (x_i^1, \ldots, x_i^d)$ and $x_j = (x_j^1, \ldots, x_j^d)$.

And the metric for SC, GGC and MOGGC which has been used in the Similarity Matrix Generation is the Radial Basis Function (RBF) defined by [77]:

$$s(x_i, x_j) = e^{-\sigma||x_i - x_j||^2} \tag{4.4}$$

The σ value has been calculated using the approximation method elaborated by Andrew Ng in [155].

Also, the genetic approaches have been initialized with different parameters. The best parameters for the GAs are shown in Tables 4.1 and 4.6 for GGC and Tables 4.2 and 4.7 for MOGGC. These parameters have been chosen from the experimental results as the best convergence parameters found, nevertheless, other parameter choices should obtain similar results. Finally, the K_{size} value for MOGGC can be found in Table 4.10.

4.2.4 Synthetic results for the MOGGC algorithm

Figure 4.2 shows the classification results of the different datasets. Table 4.1 and Table 4.2 show the best fitness values achieved by the GGC and MOGGC algorithms respectively and the parameters selection. In these cases, the σ parameter to generate the Similarity Graph of the Spectral Clustering, GGC and MOGGC algorithms is 100 (it has been approximated using the method described by Ng et al. [155]). The best accuracy results have been selected for the algorithms.

MOGGC and GGC correctly classify *Aggregation*. GGC achieves a fitness value of 0.99 which is the maximum value of fitness achieved by the algorithm (it might be a consequence of those elements which could belong to two clusters) and MOGGC achieves a fitness value of 1.0 for both, Data Continuity Degree and Clusters Separation which means that the continuity of the information of each clusters is high and also the differences between the clusters. EM, K-means and SC have problems related to the form of the data. These problems could be a consequence of local minimum convergence in the search space.

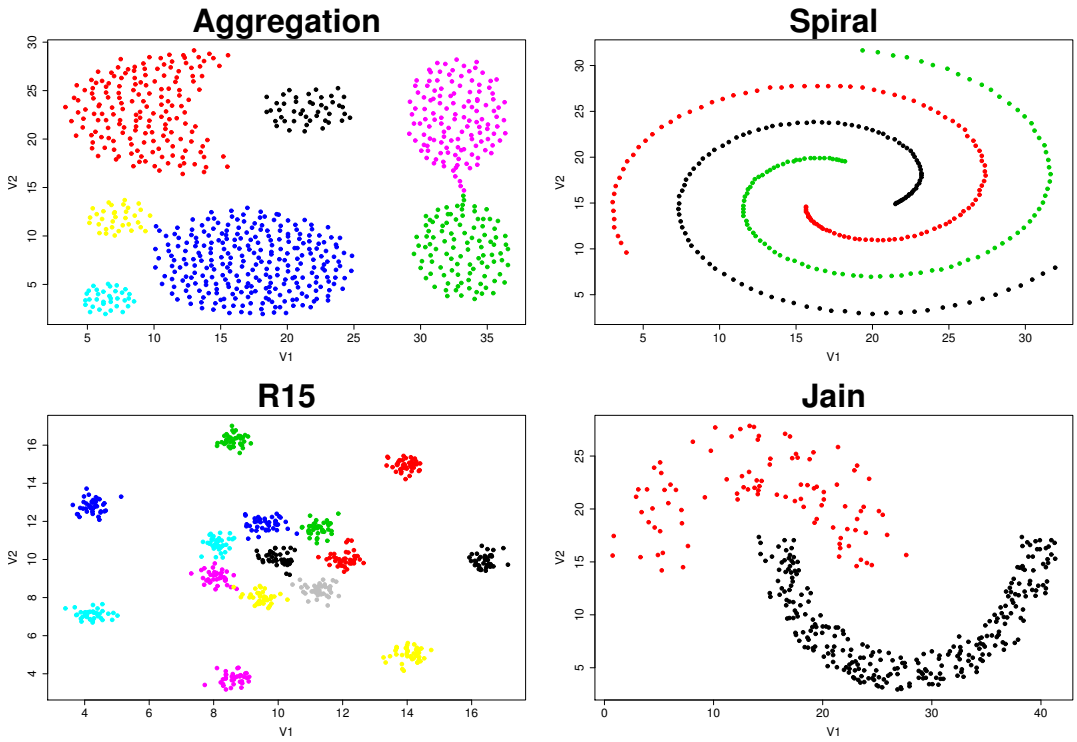


Figure 4.2: Results of **GGC** and **MOGGC** algorithms. From left to right and from top to bottom: "Aggregation", "Spiral","R15" and "Jain".

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit. DC	Fit. CS
Ag	1000	200	0.1	0.01-10^{-4}	50	1.0	1.0
Jn	100	200	0.4	0.03-10^{-4}	10	1.0	1.0
R15	100	2000	0.4	0.01-10^{-4}	50	0.87	0.98
Sp	1000	200	0.4	0.03-10^{-4}	50	1.0	1.0

Table 4.2: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in MOGGC algorithm for the different real datasets and the best fitness value obtained. The tournament size is 7.

Data	K-means	EM	SC	GGC	MOGGC
Ag	86.29%	78.68%	88.66%	▲**100%**	▲**100%**
Jn	78.28 %	56.83%	**100%**	**100%**	**100%**
R15	80.50 %	99.66%	81.33%	▲**100%**	▲**100%**
Sp	34.61 %	34.93%	**100%**	**100%**	**100%**

Table 4.3: Best accuracy values obtained by each algorithm using the synthetic datasets. The ▲symbol shows when the Wilcoxon test value is lower than 0.05.

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–EM	1	9.95	FALSE
K-means–SC	7	9.95	FALSE
K-means–GGC	11	9.95	TRUE
K-means–MOGGC	11	9.95	TRUE
EM–SC	6	9.95	FALSE
EM–GGC	10	9.95	TRUE
EM–MOGGC	10	9.95	TRUE
SC–GGC	4	9.95	FALSE
SC–MOGGC	4	9.95	FALSE
GGC–MOGGC	0	9.95	FALSE

Table 4.4: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets.

Spirals is impossible to classify using parametric algorithms and the Euclidean distance (in this case, K-means and EM). This dataset is a perfect example for continuity-cluster separation algorithms such as SC, GGC or MOGGC, for that reason, all of them achieve the best accuracy values (see Table 4.3) with the highest fitness in GGC and MOGGC (see Tables 4.1 and 4.2).

Jain is also difficult for parametric techniques. It produces low accuracy values for EM and K-means compared with SC, GCC and MOGGC. This dataset is usually used to test continuity-clustering algorithm modifying the density of the clusters, in this case, the first clusters has a clearly lower data density than the second. These non-parametric algorithms (including SC) have been designed to deal with this kind of problems as the accuracy and fitness results shows.

The *R15* dataset is a good election to test MOGGC algorithm as a parametric algorithm. The results for this dataset shows that EM obtains the best results from classical algorithms. SC obtains worse results than EM due to the noisy information of the clusters, which cover the center of the image (see Figure 4.2). MOGGC and GGC obtain the maximum accuracy values, however, the fitness values are lower than in the previous problems (see Tables 4.1 and 4.2). GGC fitness value might be a consequence of the noisy information because the clusters are closed to each other. That should be the reason because MOGGC has lower cluster separation (0.98). The continuity degree is also lower (0.87) than in the other cases, it must be because some instances of the clusters are separated from the rest.

The Friedman test applied to the algorithms proves that there is significantly difference (the p-value is 0.0123). The multiple comparison (see Table 4.4) shows that K-means and EM are Significantly different to GGC and MOGGC, while SC has not significantly different. Applying the Wilcoxon test per dataset, we find that there is significantly different for Aggregation and R15 (see Table 4.3).

This synthetic analysis gives some intuitions about the effectiveness of the MOGGC algorithm and the similarity between its results and GGC results. The following experiments will check the efficiency on real-world datasets.

Data	I. Attributes	F. Attributes	I. Elements	F. Elements
LM	90	18	360	360
OL	73	28	2536	1867

Table 4.5: Datasets reduced in the preprocessing process. This table shows the Initial Attributes and Elements, and the Final Attributes and Elements after the reduction process.

4.2.5 Real-world results for the MOGGC algorithm

This section shows the results of the MOGGC algorithm applied to real world datasets. First, it is focused on the preprocessing phase of the datasets. Next, the experimental results obtained from MOGGC are compared against to the classical algorithms considered and GGC.

4.2.5.1 Preprocessing

The preprocessing process follows the same steps than in Section 3.5.2.2. Table 4.5 shows the reduction results for the datasets. The reduction process has been based on the elimination of attributes with high correlations and elements with missing values.

Iris (Ir), *Wine* (Wn), *Glass* (Gl), *Wine Quality* (WQ) and *Page Block* (PB) datasets contain a few number of attributes. After the analysis of the variables, the correlation shows that the dimensionality reduction is not necessary. However, in the case of *Libras Movement* (LM) and *Ozone Level Detection* (OL) datasets there are a lot of attributes which do not contribute to the analysis due to the high correlation between them (see Table 4.5). These attributes have been reduced in the first step leaving 18 of 90 attributes for Libras Movement and 28 of 73 for Ozone Level Detection. In the Ozone Level Detection datasets, there are several instances which contain missing values, all these instances have been omitted for the analysis (see Table 4.5).

All the attributes from the datasets considered have been normalized applying the techniques explained in Section 2.3.

4.2.5.2 Results

The experiments have followed the same procedure that was made in the synthetic datasets experiments. The value of σ has been approximated to 100 for *Iris*, *Wine* and *Glass*, 2 for *Libras Movement* and *Ozone Level Detected* and 0.1 for *Wine Quality* and *Page Block*. The results are shown in Table 4.8.

The results for the *Iris* dataset show that EM is the best classifier (with an accuracy of the 96,67%), MOGGC is the second (96%) whereas the GGC algorithm is the third (92%), it could be due to *Iris* dataset has instances of different classes which are closed to each other, the GGC algorithm has problems to discriminate the boundary of the clusters specially when there are intersections between the clusters. This problem also affects to MOGGC algorithm. The fitness achieved by GGC and MOGGC is high in both cases (see Table 4.6 and 4.7), it means that the solution of the algorithm should be the best solution they are able to find.

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit.
Ir	1000	2000	0.1	0.8-10^{-4}	50	0.99
Wn	100	20000	0.4	0.01-10^{-4}	50	1
Gl	100	2000	0.4	0.01-10^{-4}	10	0.70
LM	100	2000	0.01	0.01-10^{-4}	10	0.92
OL	100	200	0.4	0.01-10^{-4}	10	0.93
WQ	1000	2000	0.4	0.01-10^{-4}	10	0.80
PB	100	20000	0.4	0.01-10^{-4}	50	0.92

Table 4.6: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in GGC algorithm for the different real datasets and the best fitness value obtained. The tournament size is 2.

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit. CD	Fit. CS
Ir	1000	2000	0.1	0.1-10^{-4}	50	0.99	0.99
Wn	1000	2000	0.3	0.1-10^{-4}	50	0.89	1
Gl	100	100	0.5	0.01-10^{-4}	10	0.83	0.70
LM	100	200	0.5	0.01-10^{-4}	10	0.91	0.65
OL	100	200	0.5	0.01-10^{-4}	10	0.92	1
WQ	100	20000	0.4	0.01-10^{-4}	50	0.88	0.99
PB	1000	10000	0.4	0.2-10^{-4}	50	0.43	1

Table 4.7: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in MOGGC algorithm for the different real datasets and the best fitness values obtained. The tournament size is 7.

The results for the *Wine* dataset shows that all the algorithms obtain high accuracy values (higher than the 95%), and the Genetic Algorithms obtain a perfect classification with the maximum fitness value for GGC and maximum Cluster Separation for MOGGC. These results are a consequence of the data distribution, the classes are clearer separated than in the *Iris* case (as the different clustering techniques show). It improves the results of the GGC and MOGGC algorithms, because the boundary is clearer.

Glass dataset is a difficult classification case, the results show that both, the classical and the new algorithms have problems to blindly separate the classes. In this case, SC obtains the best classical algorithm results while GGC obtains the same value of SC and MOGGC obtains the best results. However, the fitness metric values are low for both which means that they might find other solutions in the search space although these solutions are those with higher fitness of the experimental tests.

Libras Movements dataset is also a difficult classification case, again the classical and the new algorithms have problems to blindly separate the classes. In this case, SC obtains the best results from the classical algorithms while GCC and MOGGC obtain the same results. However, the fitness metric values are still low for both.

Ozone Level Detected is easier for the continuity-clustering algorithms. In this case, SC, GCC and MOGGC obtain the best classification results.

Data	K-means	EM	SC	GGC	MOGGC
Ir	89.33%	**96.67**%	89.33%	▲92%	▲*96.00*%
Wn	95.50%	*97.19*%	95.50%	**▲100**%	**▲100**%
Gl	45.79%	*47.20*%	*47.20*%	*47.20*%	**47.66**%
LM	*46.94*%	43.61%	46.11%	**▲50.00**%	**▲50.00**%
OL	76.06%	60.15%	*94.38*%	**▲96.46**%	**▲96.46**%
WQ	23.64%	*28.50*%	**40.08**%	**40.08**%	**40.08**%
PB	45.30%	56.97 %	**75.15**%	**75.15**%	**75.15**%

Table 4.8: Best accuracy values obtained by each algorithm during the experimental results applied to the UCI datasets. The ▲symbol shows when the Wilcoxon test value is lower than 0.05.

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–EM	6	10.45	FALSE
K-means–SC	8	10.45	FALSE
K-means–GGC	16.5	10.45	TRUE
K-means–MOGGC	19.5	10.45	TRUE
EM–SC	2.0	10.45	FALSE
EM–GGC	10.5	10.45	TRUE
EM–MOGGC	13.5	10.45	TRUE
SC–GGC	8.5	10.45	FALSE
SC–MOGGC	11.5	10.45	TRUE
GGC–MOGGC	3	10.45	FALSE

Table 4.9: Multiple Comparison of Friedman Test for the algorithms applied to the real-world datasets.

Data	Instances	K_{size}	GGC/SC SimMat	MOGGC SimMat
Ag	788	7	4.7 MB	0.04 MB
Jn	373	10	1 MB	0.03 MB
R15	1500	9	17 MB	0.1 MB
Sp	312	4	0.75 MB	0.01 MB
Ir	150	15	0.16 MB	0.02 MB
Wn	178	8	0.24 MB	0.01 MB
Gl	214	9	0.32 MB	0.01 MB
LM	360	9	0.99 MB	0.02 MB
OL	1867	9	27 MB	0.13 MB
WQ	4898	10	183 MB	0.37 MB
PB	5473	10	229 MB	0.42 MB

Table 4.10: Storage GGC and MOGGC. In this case it is supposed that the Similarity Matrix is a matrix of `double` variables whose size is 8 Bytes

Wine Quality is a difficult problem for clustering techniques. The worst results are achieved by the parametric algorithms (the accuracy is lower than the 30%). The results of the non-parametric techniques are the same. MOGGC has a high value of the Clusters Separation fitness, which means that this solution is closed to the best solution that it is able to find. In the case of GGC, the fitness is 0.80, it means that the algorithm might be able to find others solutions although this value was the best convergence value reached by the algorithm.

Page Block is also a difficult problem for parametric approximation and a good memory efficiency example. The parametric algorithms have achieved low accuracy results while SC and MOGGC have achieved the same solutions. These results show that GGC and MOGGC have achieved the best solution according to the Cluster Continuity metric, however, in this case, the Continuity Degree Value is smaller than usual. Analysing the parametric clustering results, they show that the data instances within the clusters should be separated between them instead of have a clear union between them.

Finally, the Friedman test shows that there is significantly difference (the p-value is 0.002501). Table 4.9 shows the Multiple Comparison among the different algorithms. There is significantly difference among K-means, EM and GGC, MOGGC and SC and MOGGC.

4.2.6 The memory optimization of the Similarity Graph

Table 4.10 shows how the new MOGGC algorithm improves the storage of the Similarity Graph related to the SC, GGC and MOGGC algorithms. There are some cases where the memory efficiency is highly relevant such as *Ozone Level Detection*, *Wine Quality* and *Page Block*. Specially, in the *Page Block* problem, the matrix takes up the 0.2% of the original matrix. This important improvement joined with the performance efficiency, makes the algorithm highly competitive against other approaches.

4.3 Conclusions of MOGGC Algorithm

MOGGC uses a simple encoding and GA-based operations combined with the SPEA2 algorithm. In comparison to GGC, the new algorithm requires less memory while, at the same time, increases the quality of the evolved clusters. MOGGC is applied to a reduced version of the Similarity Graph which is generated in the first step of the Spectral Clustering algorithm. The results show that the new algorithm obtains excellent results that are better than the classical algorithms, and has a similar (or better) classification results than previous obtained using GGC, while the memory usage is clearly optimized.

One of the most interesting improvements that could be made to the MOGGC algorithm is to automatically select the number of clusters, next section presents a new approach that tries to deal with this issue.

4.4 Adapting the Number of Clusters

The main drawback of MOGGC is the need of a priori knowledge about the number of clusters, k, which limits the applications of the algorithm.

The following sections present the Co-Evolutionary Multi-Objective Genetic Graph-based Clustering (CEMOG) algorithm. The contribution of CEMOG is the development of a new partitional clustering algorithm that solves the k-determination problem of MOGGC. To this end, CEMOG uses co-evolution to simulate variable-length chromosomes in a Genetic Algorithm. In this way, the value of k is introduced in the evolutionary search and eventually the Pareto Front provides a set of k corresponding to the trade-off of the solutions.

4.5 The Co-Evolutionary Multi-Objective Genetic Graph-based Clustering (CEMOG) Algorithm

This section describes the Co-Evolutionary Multi-Objective Genetic Graph-based Clustering (CEMOG) algorithm. CEMOG is a continuity-based clustering algorithm that was created using MOGGC [133] as a starting point. MOGGC was created to improve the robustness of the solutions reducing the dependency to the metric parameters and the search space. The main improvement of this new algorithm, compared with MOGGC, is that it is not necessary to give an initial number of clusters.

This approach combines MOGA with two objectives to guide the heuristic search using a co-evolutionary structure. CEMOG is applied in three steps:

1. **Similarity Graph generation:** A Similarity Function (usually based on a kernel function) is applied to the data instances, connecting all the points with each other. It generates the Similarity Graph.
2. **Genetic search:** CEMOG uses a MOGA to find a good graph partition. Giving an initial range of possible number of clusters $[k_{min} - k_{max}]$, the MOGA generates an initial

population, with a sub-population per k value, of possible solutions and evolves them using a fitness function to guide the algorithm to find the best solution. It stops when a good solution is found, or a maximum number of generations is reached.

3. **Clustering association:** The best solution of the Pareto Front is chosen as a solution of the algorithm and the data instances are assigned to the clusters according to the solution chosen and the sub-population who has generated this solution. The selection criterion is explained in Section 4.5.6.

4.5.1 Encoding

The encoding is a simple label-based representation [90] that follows the classical integer representation of GAs. Each individual is a n-dimensional vector (where n is the number of data instances) which has integer values between 1 and the number of clusters of the sub-population it belongs. Each individual represents a cluster selection of the dataset.

4.5.2 The k-adaptive approach

The design which helps to achieve the k-adaptive number of clusters goal is a co-evolutionary approach combined with a multi-objective algorithm. The co-evolution is focused on two points of view which are described below: macro-evolution (i.e., the evolution of the whole population) and micro-evolution (i.e., the evolution of each sub-population).

4.5.3 Macro-evolution and the exchanger operator

We use a graph topology for migrating individuals from sub-populations. The assumption is that all sub-populations have the same representation and same goals to solve (see Figure 4.3).

Each sub-population represents a possible k-value ranged from k_{min} to k_{max}. These values generate a higher search space which complicates the genetic search; nevertheless, the new methodology helps to find a satisfactory solution in a range of possible number of clusters instead of a fixed one. The algorithm looks for the solutions in the different sub-populations. Moreover, it also uses a exchanger to send individuals from a population to another, modifying the environment of the different sets of chromosomes and encouraging genetic diversity. This exchanger improves the quality of the solutions and reduces the local solution convergence (e.g., local minimum). The exchanger developed in CEMOG exchanges two random couples of each sub-population with its neighbour sub-populations, i.e., a couple of individuals of population $SPop_n$ is send to $SPop_{n+1}$ and other couple to $SPop_{n-1}$ (see Figure 4.3).

4.5.4 Micro-evolution and the MOGA operators

On the one hand, CEMOG, as a MOGA, uses the SPEA2 algorithm for the genetic evolution of the set of solutions within the sub-populations. On the other hand, as a clustering algorithm, CEMOG begins with a K_{size}-Similarity Graph in the same way that the Spectral Clustering

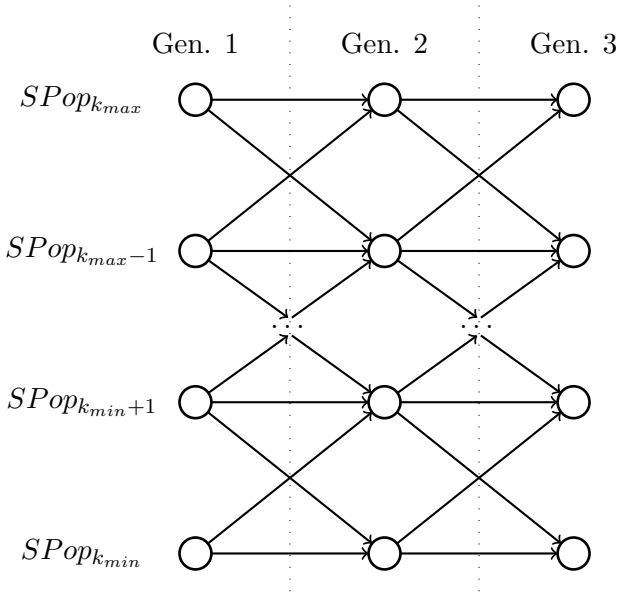


Figure 4.3: Once the range of k values is set between k_{min} and k_{max}, the set of sub-populations is generated for each k value. The different individuals can be moved between sub-populations in each generation.

algorithm [193]. The K_{size} value limits the memory used to a matrix $K_{size} \times N$ where N is the number of data instances.

Finally, the MOGA operators used are the same that MOGGC (see Section 4.1.1).

4.5.5 The fitness objectives

The fitness function are the same two objectives of MOGGC: improve the data continuity degree and cluster separation. For that reason, we have use the metrics of MOGGC (see Section 4.1.2):

- Data Continuity Degree.
- Cluster Separation.

4.5.6 Choosing the solution from the Pareto Front

Due the necessity to choose one of the solutions from the Pareto Front, the experimental results show that the solution with the highest value of the cluster separation metric in the Pareto Front always obtains better accuracy values compared with human-based classification. Therefore, this value has been chosen as the algorithm solution.

	SPop Size	k_{min}	k_{max}	Gen.	Cross.	Mut.	Eli.
Ag	200	5	9	500	0.1	0.01-10$^-$4	10
Jn	50	2	6	200	0.2	0.2-10$^-$4	5
R15	50	13	17	500	0.2	0.3-10$^-$4	5
Sp	200	2	6	200	0.1	0.01-10$^-$4	10

Table 4.11: Best parameter selection (SubPopulation size, Generations, Crossover probability, Mutation probability and Elitism size) used in CEMOG algorithm for synthetic datasets and the best fitness value obtained. The tournament size is 2.

4.6 Experimental Results

This section shows the results of the experiments that have been carried out to assess the performance of CEMOG to find good k-values and compare its performance with similar algorithms. The first part presents the datasets which have been used to test the algorithm. The second one describes the evaluation metrics and the experimental set-up. Finally, the last part shows the results on the synthetic and real-world datasets which have been taken from the literature.

4.6.1 Synthetic results for the CEMOG algorithm

Tables 4.11 and 4.12 show the parameters selection for MOGGC and CEMOG, respectively. These parameters have been chosen after a deep search of parameters in several ranges. In these cases, the σ parameter to generate the Similarity Graphs of MOGGC and CEMOG is 100 (it has been approximated using the method described by Ng et al. [155]). The best accuracy results were selected for the algorithms. Table 4.13 shows the comparison of all the algorithms considered.

Figure 4.4 depicts the Pareto Front of CEMOG applied to the synthetic datasets. Figure 4.5 shows the accuracy values for different sub-populations of each dataset. The criterion used to choose these solutions prioritizes the cluster separation objective in the Pareto Front considering the solution of the data continuity objective.

CEMOG and MOGGC correctly cluster the Aggregation dataset. The Pareto Front defined by MOGGC (see Figure 4.4) shows a dominant solution (mark with 'x' at the top-rigth corner). This winner solution belongs to the $k = 7$ sub-population which is the solution with the best accuracy (see Figure 4.5). EM, K-means and SC have problems related to the data form. These problems might be a consequence of local minimum convergence in the search space.

The Spirals dataset is impossible to cluster using parametric algorithms and the Euclidean distance (that is, K-means and EM). This dataset is a perfect example for continuity-cluster separation algorithms such as SC, MOGGC or CEMOG. For that reason, all of them achieves the best accuracy values (see Table 4.13). Furthermore, analysing the Pareto Front defined by CEMOG for Spirals, there is only one dominant solution (marked with 'x' at the to right corner), corresponding to $k = 3$ sub-population, which is the sub-population with the best accuracy (see Figure 4.5).

The Jain dataset is also difficult for parametric clustering. It produces low accuracy values for EM and K-means compared with SC, MOGGC and CEMOG. This dataset is usually used to

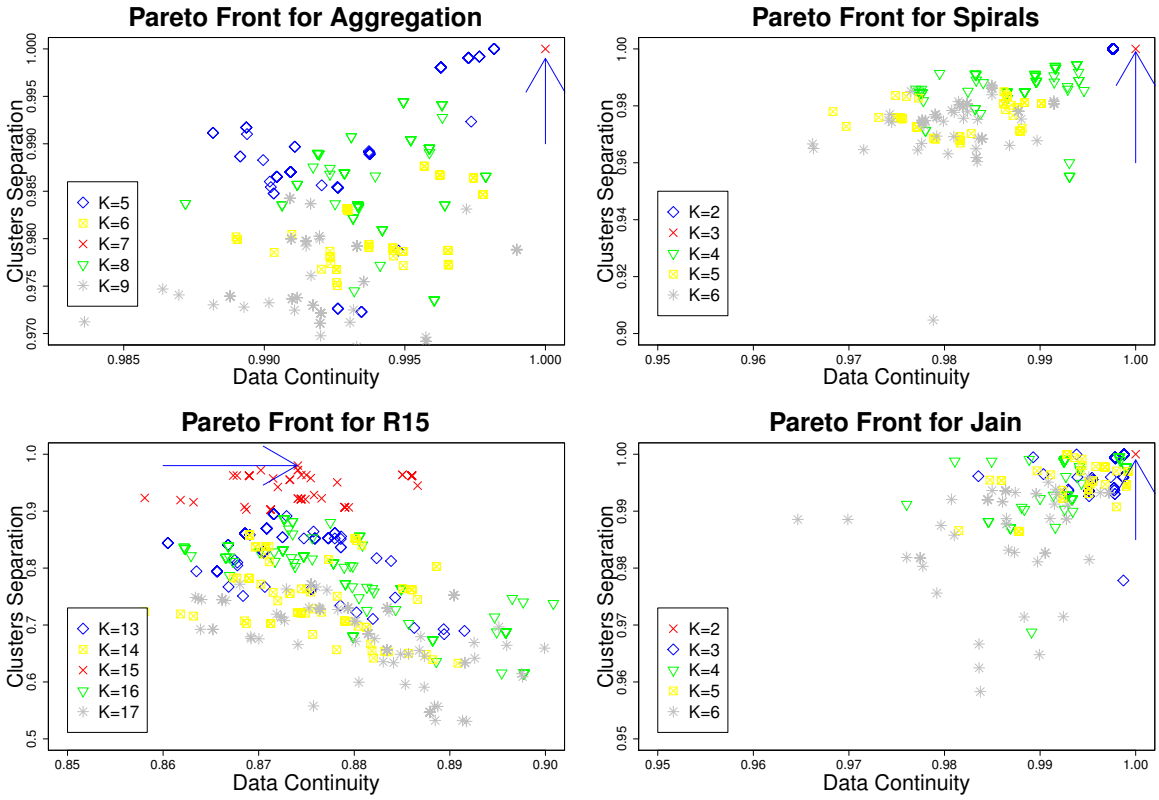


Figure 4.4: Pareto Front generated by the CEMOG sub-populations chosen for the synthetic datasets. From top to bottom: "Aggregation", "Spiral","R15" and "Jain". The arrows mark the best solution.

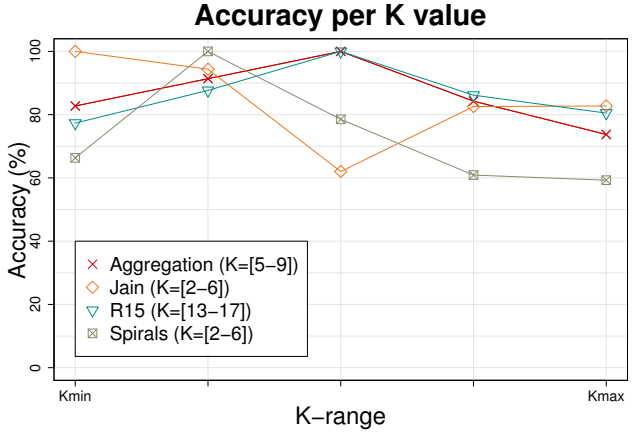


Figure 4.5: Accuracy results for CEMOG algorithm ranged from k_{min} to k_{max}. The legend shows the values for the range per synthetic dataset.

Data	Pop.	Gen.	Cross.	Mut.	Eli.
Ag	1000	200	0.1	0.01-10^{-4}	50
Jn	100	200	0.4	0.03-10^{-4}	10
R15	100	2000	0.4	0.01-10^{-4}	50
Sp	1000	200	0.4	0.03-10^{-4}	50

Table 4.12: Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in MOGGC algorithm for the different synthetic datasets and the best fitness value obtained. The tournament size is 7.

Data	K-means	EM	SC	MOGGC	CEMOG
Ag	86.29%	78.68%	88.66%	▲**100**%	▲**100**%
Jn	78.28 %	56.83%	**100**%	**100**%	**100**%
R15	80.50 %	99.66%	81.33%	▲**100**%	▲**100**%
Sp	34.61 %	34.93%	**100**%	**100**%	**100**%

Table 4.13: Best accuracy values obtained by each algorithm using the synthetic datasets. The ▲symbol shows when the Wilcoxon test value is lower than 0.05.

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–EM	1	9.95	FALSE
K-means–SC	7	9.95	FALSE
K-means–MOGGC	11	9.95	TRUE
K-means–CEMOG	11	9.95	TRUE
EM–SC	6	9.95	FALSE
EM–MOGGC	10	9.95	TRUE
EM–CEMOG	10	9.95	TRUE
SC–MOGGC	4	9.95	FALSE
SC–CEMOG	4	9.95	FALSE
MOGGC–CEMOG	0	9.95	FALSE

Table 4.14: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets.

test continuity-clustering algorithms modifying the density of the clusters, in this case, the first cluster has clearly lower data density than the second cluster. According to the Pareto Front, CEMOG also has a clear dominant solution (marked with 'x' at the top-right corner) which is the k value with best accuracy. MOGGC and SC have also good results.

In the case of the R15 dataset, the experiments show that EM obtains the best results for classical algorithms. SC obtains worse results than EM due to the noisy information and, therefore, the boundaries are not clearly defined. MOGGC obtains the maximum accuracy value. In the CEMOG case, the Pareto Front is defined by several solutions which cover different k values (see Figure 4.4). In order to define the best solution, the cluster separation objective has been prioritized over the data continuity objective, as was mentioned before. Applying this criterion, the solution chosen is the top 'x' instead of the top right. The accuracy value of the solution is the maximum as Figure 4.5 shows.

The Friedman test shows that the p-value is 0.0123 which means that there is significantly difference. The Multiple Comparison (see Table 4.14) shows that K-means and EM are significantly different to GGC and MOGGC. The Wilcoxon test comparing SC with MOGGC and CEMOG per dataset shows that there is significantly differente in 2 datasets for CEMOG and MOGGC (see Table 4.13).

Finally, Figure 4.5 shows an interesting remark. The k-values close to the best accuracy values have generally more accuracy than the rest, except for Jain, i.e., it seems that the distance of k to the optimal k is correlated with its accuracy. This observation suggests a way to automatically set the k_{min} and k_{max} values.

4.6.2 Real-world results for the CEMOG algorithm

This section shows the experimental results of the CEMOG algorithm applied to real world datasets. We fist describe the datasets preprocessing followed by the experimental results. As in the previous section, CEMOG is compared against the classical clustering algorithms (K-means, EM and SC) and MOGGC.

4.6.2.1 Algorithm execution

The experiments followed the same procedure than the previous synthetic datasets experiments. The value of σ was approximated to 100 for *Glass*, 2 for *Libras Movement* and *Ozone Level Detected* and 0.1 for *Wine Quality* and *Page Block*. The results and best k value for CEMOG are shown in Table 4.16. Following, the results for each dataset are described.

Glass dataset is difficult for clustering; the results show that both, the classical and the new algorithms have problems to blindly separate the classes. In this case, SC obtains the best classical algorithms results. MOGGC obtains good results; however, CEMOG obtains the best results. A remarkable fact is that the k selection process has chosen k = 5 while the original number of classes is 7. This might explains why the other clustering algorithms have worse results than CEMOG.

Libras Movements dataset is also a difficult classification case, again the classical and the new algorithms have problems to blindly separate the classes. In this case, K-means obtains

Data	SPop.	k_{min}	k_{max}	Gen.	Cross.	Mut.	Eli.
Gl	100	5	9	100	0.5	0.01-10^{-4}	10
LM	100	13	17	200	0.5	0.01-10^{-4}	10
OL	100	2	6	200	0.5	0.01-10^{-4}	10
WQ	100	5	9	2000	0.4	0.01-10^{-4}	50
PB	1000	3	7	1000	0.4	0.2-10^{-4}	50

Table 4.15: Best parameter selection (Sub-Population, Generations, Crossover probability, Mutation probability and Elitism size) used in CEMOG algorithm for the different real datasets and the best fitness values obtained. The tournament size is 7.

Data	K-means	EM	SC	MOGGC	CEMOG
Gl	45.79%	*47.20%*	*47.20%*	**47.66%**	▲**52.34%** (k=5)
LM	*46.94%*	43.61%	46.11%	▲**50.00%**	▲48.05% (k=14)
OL	76.06%	60.15%	*94.38%*	▲**96.46%**	▲**96.46%** (k=2)
WQ	23.64%	*28.50%*	**40.08%**	**40.08%**	**40.08%** (k=7)
PB	45.30%	56.97 %	**75.15%**	**75.15%**	**75.15%** (k=5)

Table 4.16: Best accuracy values obtained by each algorithm during the experimental results applied to the UCI datasets. The ▲symbol shows when the Wilcoxon test value is lower than 0.05.

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–EM	0.5	10.95	FALSE
K-means–SC	7.5	10.95	FALSE
K-means–MOGGC	13.5	10.95	TRUE
K-means–CEMOG	13.5	10.95	TRUE
EM–SC	7.0	10.95	FALSE
EM–MOGGC	13.0	10.95	TRUE
EM–CEMOG	13.0	10.95	TRUE
SC–MOGGC	6.0	10.95	FALSE
SC–CEMOG	6.0	10.95	FALSE
MOGGC–CEMOG	0	10.95	FALSE

Table 4.17: Multiple Comparison of Friedman Test for the algorithms applied to the real-world datasets.

the best results from the classical algorithms. MOGGC obtains the best general results and CEMOG obtains similar results than MOGGC because the k selection has chosen a different k value of the number of classes.

Ozone Level Detected is easier for the continuity-clustering algorithms. In this case, SC, MOGCC and CEMOG obtain the best classification results, all with the same accuracy value.

Wine Quality is a difficult problem for clustering techniques. The worst results are achieved by the parametric algorithms (the accuracy is lower than the 30%). The results of the non-parametric techniques are the same.

Page Block is also a difficult problem for the parametric approximation. These algorithms have achieved low accuracy while SC, MOGGC and CEMOG have achieved the same solutions, with better accuracy. Analysing the parametric clustering results, they show that the data instances within the clusters should be separated between them instead of having a clear union between them.

The Friedman test p-value is 0.003535, which means that, at least, there are two algorithms which are different. The Multiple Comparison (see Table 4.17) shows that there is significantly difference among K-means and EM compared against GGC and MOGGC. The Wilcoxon test comparing SC with MOGGC and CEMOG per dataset shows that there is significantly differente in 3 datasets for CEMOG and 3 for MOGGC (see Table 4.16).

Due to SC, MOGGC and CEMOG are focused on continuity-based clustering, it might be the reason because these algorithms achieve similar results. The main advantage of CEMOG compared against the others is that it does not need a fixed number of clusters and also find more solutions which can be promising.

4.7 Conclusions of CEMOG

CEMOG presents a new k-adaptive graph-based clustering algorithm, inspired by MOGGC, that combines Co-Evolution with Multi-Objective Genetic Algorithms. CEMOG uses a simple integer encoding in a GA combined with the SPEA2 algorithm. In comparison to MOGGC, the new algorithm is a k-adaptive approach,which obtains good results in a bigger search space. The results show that the new algorithm obtains excellent results that are better than the classical algorithms, and has a similar (or better) clustering results than previous obtained using MOGGC.

Chapter 5

Genetic Graph-based Clustering for Streaming Data Analysis

"Welcome to the Island of Misfit Toys."

- Stephen Chbosky

Previous chapters have studied how the genetic graph-based algorithms have been applied to different datasets (real and synthetic). However, these datasets are small (and static) compared with those others generated by real-time data flows. Usually, real-time data flows generate set of points which should be analysed as fast as possible (the ideal approach would be a real-time analysis). This new perspective extends the possibilities to other fields such as Large or Stream Analysis.

The Stream Data, or Data Stream Model, is usually defined by the following features:

- **The data elements arrive online.** It means that the algorithm is directly connected to a data stream and it is reading and processing information in real-time.
- **The order of the data is uncontrollable.** This modifies the trends produced in the patterns which can be extracted from the data.
- **The dataset is usually very large.** The information that the algorithm stores about the data is limited.
- **Once an element is processed it is discarded or archived.** This supposses that the algorithm needs to be ready to process each data instance only once.

Examples of these datasets can be found in Social Networks, the Internet or telephone records, data logs, etc.

This chapter extends our previous algorithms to introduce two new approaches which have been created to deal with Large and Stream Data in two different ways: offline and online.

5.1 Dealing with Large and Stream Data Offline

Usually when we are working with large data, such as stream data (which is quite common when we are working with telephone records, sensors, etc), we can consider two different approaches. The first one is based on the information that we can extract from the total amount of data, and the second one is the information that we can extract while the data is coming. This first algorithm which has been designed in this work is focused on the former. First, we generate a set of centroids; this set of centroids is optimized applying a centroid-based algorithm, such as K-means. Later, the algorithm groups those centroids, and the regions represented by them, using a graph-based approach which considers these regions as nodes in the graph and tries to join them according to their similarity. In order to understand the whole process it is important to consider the information which is available in the search space. Following this idea, this chapter has been focused on theoretical clustering analysis, to better understand it, the chapter provides a brief introduction to Voronoi Tessellation.

5.1.1 The Voronoi Tessellation

Let $X = \{x_1, \ldots, x_n\}$ be a dataset with n elements. As it was mentioned before, clustering is the process to blindly group these points into j clusters called $C = \{c_1, \ldots, c_j\}$. Some clustering techniques use centroids to define the clusters, therefore we can define $V = \{v_1, \ldots, v_j\}$ as the centroid set associated to a clustering algorithm. Typically, $j << n$. We can define the Voronoi region R_i of the centroid v_i as the sets of points in $\mathbb{R}^d$ for which v_i is the nearest point:

$$R_i = \{z \in \mathbb{R}^d \mid i = \arg\min_j ||z - v_j||^2\}. \tag{5.1}$$

The boundaries of each Voronoi region are **linear segments**. We can define the Voronoi set as C where each c_i is:

$$c_i = \{x \in X \mid i = \arg\min_j d(x, v_j)\}. \tag{5.2}$$

The partition C is also known as *Voronoi Tessellation* or *Dirichlet Tessellation* [59].

The Voronoi Tessellation can also be helpful to determine manifolds. In our context, a manifolds can be defined as [112]:

Definition 5.1.1 (Manifold)**.** An n-dimensional manifold (n-manifold for short) is a subset of some Euclidean space $\mathbb{R}^k$ that is locally Euclidean of dimension n.

This means that whether we are close enough, the manifold behaves like an Euclidean space. The idea of the algorithm is to separate the manifold using Voronoi regions and them join these regions considering their continuity. This would reduce the computational effort when:

- **The divisions are clearly defined**, because the centroid would be a representative point in the search space, summarizing a lot of information about the region data.
- **There is an optimum way to join the regions by continuity**, because the manifold definition would be clearer.

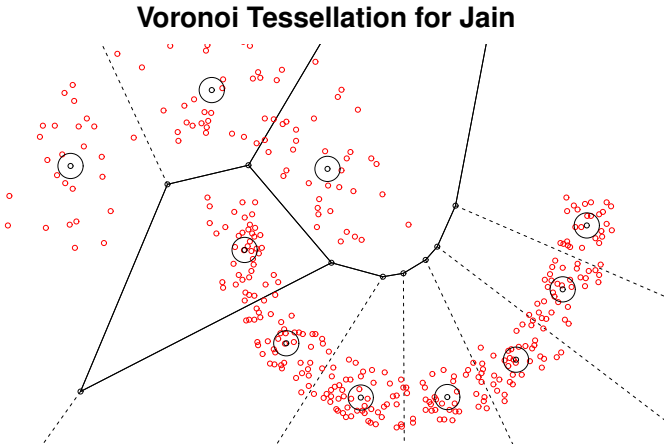


Figure 5.1: Voronoi tessellation of Jain dataset applying K-means

Figure 5.1 illustrates an example. This brief introduction about Voronoi regions gives some intuition about the search space division generated during the clustering process. Using this information we are able to structure the algorithm in several different ways. We can consider a first step (micro-search) based on a set of clusters (composed by the centroids and the Voronoi Regions) and them group these sets (macro-search) in manifolds. However, we want to keep information about the data continuity. For that reason, our first goal will be to combine MOGGC (macro-search) and K-means (micro-search) algorithms in order to generate a 2-step clustering algorithm that could be able to deal with massive and real-time data.

5.2 The Multi-Objective Genetic Graph-based Streaming Clustering Algorithm (MOGGSC)

This section describes the Multi-Objective Genetic Graph-based Streaming Clustering Algorithm (MOGGSC). This algorithm combines a clustering version of K-means implemented in MapReduce and the MOGGC algorithm in order to reduce the computationally effort and improve the scalability for large datasets of our previous algorithms.

5.2.1 Micro Search: K-means and Voronoi Regions

The micro search chooses the Voronoi Regions that the algorithm is going to use during the clustering process. These regions are chosen using a parallelizable version of K-means (see Section 2.7.1.1). The algorithm defines the regions optimizing the centroid positions. This information is sent to the next level in order to improve the clustering decision.

Using only the centroid position we are able to reduce the genetic search reducing the chromosomes size and the computationally effort during the search. The whole process has been divided in three main steps (see Figure 5.2):

1. **Ini:** First, the data is split into several data blocks. Each data block corresponds to a node or machine that we are using during the parallelization process. During this initialization process the initial centroids are chosen, and all the nodes (machines) need to have an updated copy of the centroids (see Figure 5.2 *Ini*).

2. **Map:** Using the information of the centroid positions, each node has to assign each data point to the closest centroid. This process is done during the "Map Step" of MapReduce algorithm. It assigns each data instance to a data centroid and group the data by centroid in order to send them to different nodes for the Reduce step (see Figure 5.2 *Map*).

3. **Reduce:** The data is moved to different nodes in order to calculate the new centroids position. Each node should have all the data instances which have been assigned to the centroid that they are trying to recalculate. Then, the reducer calculates the centroid positions and the list of centroids is updated and sent to all the nodes again (see Figure 5.2 *Reduce*). If the algorithm has achieved a convergence criteria or a maximum number the generations, the process stops. Instead, it goes back to step two.

The whole process can be considered equivalent to standard K-means algorithm.

5.2.2 Macro Search: MOGGC

The macro search joins previous regions in order to create good clusters. These approaches are usually used in online clustering algorithms as will be explained in the followings sections. Using the information of the centroids, we cluster the centroids in bigger clusters according to their continuity.

The manifold reconstruction process is based on the information extracted during the first step. The regions form a graph, they are used to represent the nodes, while their similarities are used to represent the edges. This topology is used to define the final clusters using a graph-cut methodology. In this work, the MOGGC algorithm has been chosen for this task (see Section 4.1).

The MOGGC algorithm has the advantage that the search can be easily parallelizable. The population can be divided among different machines and the fitness functions can be calculated by chromosomes, which implies that the set of chromosomes do not need to access to other individual information. Also, the selection divides the chromosomes in order to reproduce them among different machines (see Figure 5.2). Once the new generations have been created, the members of the population continue with this process.

This macro search concludes when the algorithm has identified reasonable clusters, or it achieves a maximum number of generations. Due to this process is only working with the regions, we can use the same encoding, reducing the chromosome size. The fitness functions are calculated using the centroids as regions representatives.

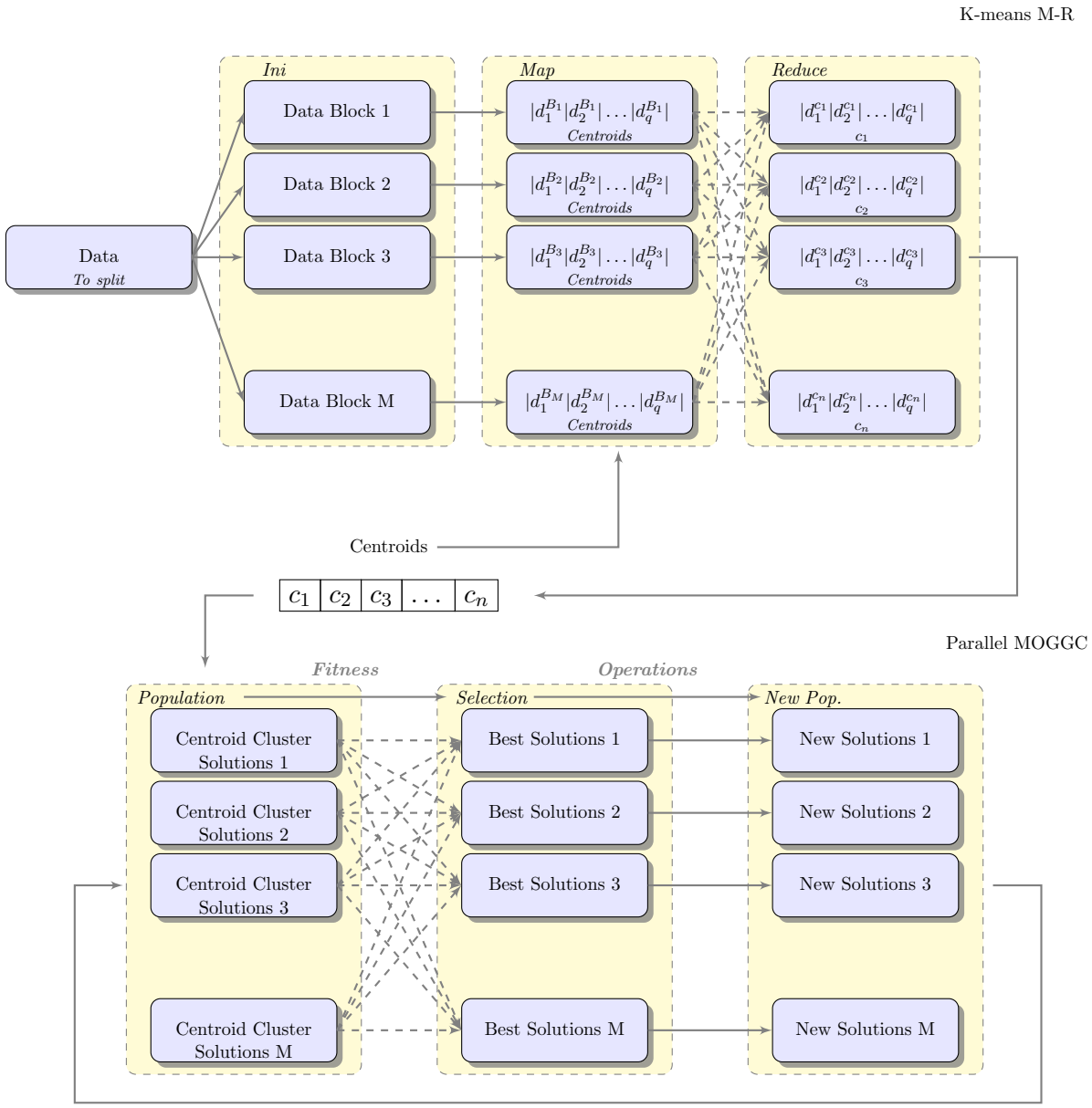


Figure 5.2: MOGGSC algorithm. The micro-search is carried out using a K-means and Map Reduce approach, while the macro-search joins the Voronoi Regions generated during the micro-search through a parallel version of the MOGGC algorithm.

5.3 MOGGSC validation

In order to evaluate the algorithm we have analysed its complexity and scalability. On the one hand, due to the clustering process uses compress information about the search space, the second step (Macro) depends on the number of centroids chosen. On the other hand, the MapReduce version for K-means is scalable and extensive, it depends on the number of clusters and the data points (and also the messages interchanged when different machines are used).

5.3.1 Algorithm Complexity: Time and Memory

Each step has a complexity order, the first step has the complexity of the clustering algorithm (in this case Big K-means [211]) which is $O(k \cdot N/p)$, where k is number of clusters, N is number of data points and p is number of parallel nodes.

The complexity of the genetic search depends on the number of chromosomes and the operations chosen. MOGGC has a similar complexity than GGC (see Section 3.4.1) except during the fitness calculation and the SPEA2 execution. The fitness depends on the number of centroids (in this case) and it is, at most, $O(c^2)$, because the operations are, at most, considering all data instances, pair to pair. SPEA2 execution depends on the population size P and the archive size $\bar{P}$. It is calculated as $O((P + \bar{P})^2 \cdot \log(P + \bar{P}))$.

Hence, the total complexity of the MOGGSC algorithm is considering G generations for the genetic search, because the two clustering steps are no parallelizable. Therefore, the complexity of this algorithm can be represented by:

$$O(k \cdot N/p + G \cdot (c^2 + (P + \bar{P})^2 \cdot \log(P + \bar{P}))) \tag{5.3}$$

The memory consumption is divided in the different machines. It can be calculated as:

$$N + p \cdot c + c \cdot K \tag{5.4}$$

where N is the number of instances p the number of Map-Reduces nodes, c the number of centres and K the number of neighbours considered in the Similarity Graph. The original *MOGGC* has a memory consumption of

$$N \cdot K. \tag{5.5}$$

Therefore, due to $c << N$ and the Map Reduce nodes are usually distributed in different machines, then the memory consumption is smaller for MOGGSC than for MOGGC.

5.3.2 Experimental Setup

The evaluation of the clustering algorithm is a sensitive process, specially when the algorithm deals with large datasets. In this work we have focused the evaluation on comparing the algorithm with other offline algorithms. This evaluation has been divided in two different steps:

- The first step is used to compare the algorithm against classical algorithms. This evaluation has been carried out over a limited number of instances, due to classical algorithms are not

designed to deal with large data problems. The size of these datasets has fixed to 10000 instances.

- The second step compares some evolutions of these classical algorithms applied to large data problems.

For the first experiments, we have chosen the classical algorithms: K-meas, EM and SC which were described in Section 2.4.

The experiments with 50.000 instances have been carried out using the Nystrom extension of Spectral Clustering [69] (SC+N) -which uses only a subset of the whole dataset; and the MapReduce version of K-means [211] (Big K-means).

The metric used for all the algorithms is the Euclidean Distance. The genetic parameters, for all the experiments, have been set to: 100 generations, 300 population, $(\mu + \lambda)$ selection, 5 elitism, 0.1 mutation, 0.5 crossover. The similarity metric used is RBF. The number of data instances for Nystrom reduction has been set 200.

All the experiments have been executed 100 times.

5.3.3 Dataset Description

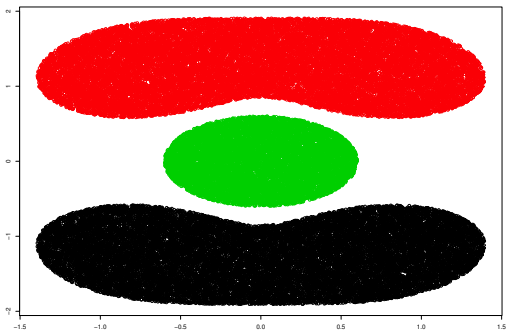
The synthetic datasets have been generated using the R package mlbench[1] which allows to generate big synthetic datasets with a topological structure. The datasets -all composed by 50000 instances- that have been generated are the following (see Figure 5.3):

- **Cassini**: This dataset is formed by 3 clusters which are continuity-based.
- **Cuboids**: This dataset has four cuboids in three dimensions.
- **Hypercube**: This datasets is composed by eight spheres distributed as the vertex of a cube.
- **Shapes**: This dataset has 4 different shapes.
- **Simplex**: This dataset has four spheres well separated.
- **Smiley**: This dataset is formed by four clusters which define a smiley face.
- **Spirals1**: This dataset has two spirals without noise.
- **Spirals2**: This dataset has two noisy spirals.

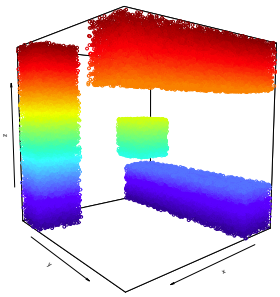
The algorithms have also been compared using real-world datasets. The datasets which have been chosen are:

- Forest Covertype Dataset (**CovType**): This dataset is from the UCI Machine Learning repository. It has been used to predict forest cover type from cartographic variables. It has 7 classes, 54 attributes and 581012 instances

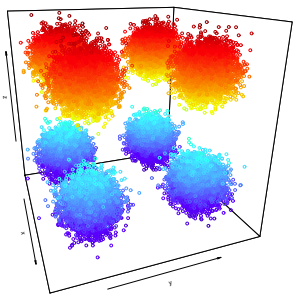
[1] `http://cran.r-project.org/web/packages/mlbench/mlbench.pdf`



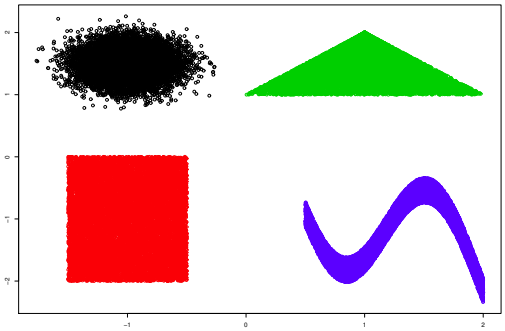
(a) Original Cassini dataset



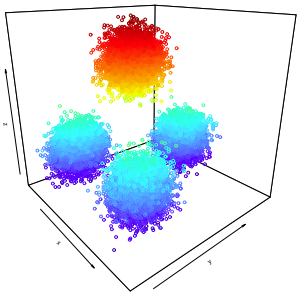
(b) Original Cuboids dataset



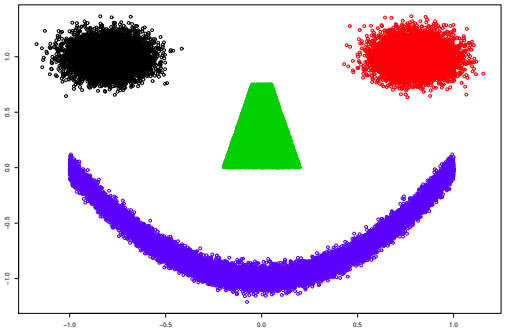
(c) Original Hypercube dataset



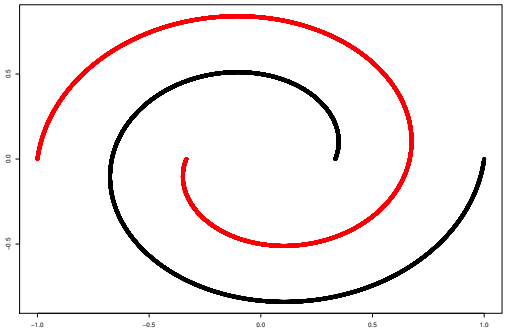
(d) Original Shapes dataset



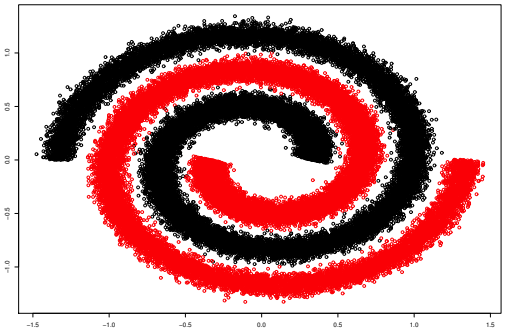
(e) Original Simplex dataset



(f) Original Smiley dataset



(g) Original Spiral1 dataset



(h) Original Spiral2 dataset

Figure 5.3: The original images of the synthetic datasets

- **Electricity**[2]: This data records different prices from electricity consumption. It is composed by 2 classes, 5 attributes and 45312 instances
- **Statlib**[3]: This dataset contains information of each stock of Dow Jones 30. It has 30 classes, 6 attributes and 138166 instances
- Records **Block** (UCI Machine Learning Repository): This dataset contains the underlying records from the epidemiological cancer registry in North Germany. It has 2 classes, 4 attributes and 574913 instances.

5.4 Experimental Results

This section presents the evaluation process of MOGGSC. The new offline clustering algorithm for data stream has been compared against: Spectral Clustering, EM and K-means using 10.000 instances; and SC using the Nystrom method and K-means using MapReduce for 50.000 instances. Also, the algorithms for massive data analysis has been applied to real-world datasets.

5.4.1 Experiments with Synthetic data

Tables 5.1 and 5.3 show the results for the previous algorithms applied to 10.000 instances and to 50.000 instances, respectively. As we can appreciate, MOGGSC obtains generally good results, but it is important to analyse the algorithm according to each dataset in order to identify its weaknesses.

In the case of **Cassini** dataset, MOGGSC obtains good results of Median and Maximum. However, EM obtains better results according to the Mean. The first reason is due the version of EM which has been used is deterministic. Also the dataset (see Figure 5.3 (a)) has two sections which can be easily defined using a Gaussian distribution and when the number of instances is lower the boundaries between these distributions are well-defined. This allows the algorithm to discriminate the clusters clearer. SC also obtains good results but it is more sensitive to noise. K-means obtains worst results, probably because the regions defined by this algorithm are less clear. The algorithms applied to 50.000 instances show that MOGGSC obtains the best results, SC+N obtains good results and Big K-means obtains similar results than the previous application.

For **Cuboids** dataset, EM obtains the best results followed by MOGGSC which obtains the same results for the Median and the Maximum but lower results (95.27% of Mean accuracy). The minimum of MOGGSC is very low (56.37%). Considering the Median values, this observed values could be a consequence of local minimum convergence. SC obtains the worst Median results (76.71%) even worse than K-means (87.24%). This should be because the algorithm have problems identifying the volume (see Figure 5.3 (b)). For 50.000 instances, MOGGSC obtains more stable results but they are generally worse according to the Mean and Median. SC+N obtains the best results according to the Median and the second according to the Mean, but the

[2] http://www.inescporto.pt/~jgama/ales/ales_5.html
[3] http://tunedit.org/repo/StatLib/numeric/dj30-1985-2003.arff

MOGGSC	Min	Max	Median	Mean	SD
Cassini	*80.89%*	**100.0%**	**100.0%**	*99.69%*	± 0.0218
Cuboids	56.37%	**100.0%**	▲**100.0%**	*95.27%*	± 0.0838
Hypercube	**100.0%**	**100.0%**	▲**100.0%**	**100.0%**	± 0.0000
Shapes	*66.13%*	**100.0%**	▲**100.0%**	*98.55%*	± 0.0500
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	*69.57%*	**100.0%**	**100.0%**	*99.31%*	± 0.0432
Spirals1	*52.42%*	**100.0%**	**100.0%**	*97.94%*	± 0.0815
Spirals2	50.91%	**99.90%**	▼*76.93%*	*74.89%*	± 0.1756
SC	Min	Max	Median	Mean	SD
Cassini	53.66%	**100.0%**	99.59%	92.13%	± 0.1446
Cuboids	35.30%	**100.0%**	76.71%	82.72%	± 0.1791
Hypercube	31.48%	83.11%	64.21%	66.37%	± 0.1431
Shapes	35.00%	**100.0%**	69.16%	74.54%	± 0.2010
Simplex	95.36%	**100.0%**	**100.0%**	99.07%	± 0.0207
Smiley	52.77%	99.59%	98.01%	86.42%	± 0.1454
Spirals1	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Spirals2	**99.77%**	*99.82%*	**99.80%**	**99.80%**	± 0.0004
K-means	Min	Max	Median	Mean	SD
Cassini	64.94%	98.87%	65.59%	69.62%	± 0.1092
Cuboids	*71.55%*	**100.0%**	87.24%	81.19%	± 0.0916
Hypercube	63.61%	**100.0%**	81.55%	83.99%	± 0.0910
Shapes	62.60%	**100.0%**	**100.0%**	88.37%	± 0.1715
Simplex	62.54%	**100.0%**	**100.0%**	91.85%	± 0.1550
Smiley	43.37%	94.60%	63.40%	71.59%	± 0.1624
Spirals1	50.00%	50.00%	50.00%	50.00%	± 0.0000
Spirals2	59.24%	59.37%	59.37%	59.31%	± 0.0007
EM	Min	Max	Median	Mean	SD
Cassini	**99.97%**	99.97%	*99.97%*	**99.97%**	± 0.0000
Cuboids	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Hypercube	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Shapes	**99.99%**	99.99%	99.99%	**99.99%**	± 0.0000
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	**99.88%**	*99.88%*	*99.88%*	**99.88%**	± 0.0000
Spirals1	50.00%	50.00%	50.00%	50.00%	± 0.0000
Spirals2	*65.34%*	65.34%	65.34%	65.34%	± 0.0000

Table 5.1: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the synthetic datasets using 10.000 instances. Values in bold shows the best results while italics shows the second.

SD shows that the algorithm is less stable than MOGGSC (0.1736 and 0.0777, respectively). K-means obtains the worse results.

Hypercube results show that EM and MOGGSC are able to discriminate the clusters

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–SC	2.5	10.3239	FALSE
K-means–EM	6.5	10.3239	FALSE
K-means–MOGGSC	13.0	10.3239	TRUE
SC–EM	4	10.3239	FALSE
SC–MOGGSC	10.5	10.3239	TRUE
EM–MOGGSC	6.5	10.3239	FALSE

Table 5.2: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets of 10000 instances.

MOGGSC	Min	Max	Median	Mean	SD
Cassini	**78.71%**	**100.0%**	**100.0%**	**98.55%**	± 0.0391
Cuboids	**72.09%**	**100.0%**	*91.39%*	**92.66%**	± 0.0777
Hypercube	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Shapes	**70.11%**	**100.0%**	**100.0%**	**98.59%**	± 0.0444
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	**65.26%**	**100.0%**	**100.0%**	**99.22%**	± 0.0460
Spirals1	*52.14%*	**100.0%**	**100.0%**	**96.98%**	± 0.0987
Spirals2	*50.91%*	**99.90%**	**76.93%**	**74.89%**	± 0.1756
SC+Nystrom	Min	Max	Median	Mean	SD
Cassini	50.45%	**100.0%**	*98.61%*	*89.25%*	± 0.1681
Cuboids	40.52%	**100.0%**	**99.74%**	*86.16%*	± 0.1736
Hypercube	*65.78%*	**100.0%**	76.71%	79.80%	± 0.1565
Shapes	37.43%	**100.0%**	68.71%	68.71%	± 0.4425
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	63.27%	*99.30%*	*74.99%*	*87.12%*	± 0.1627
Spirals1	**54.47%**	**100.0%**	**100.0%**	**97.15%**	± 0.0991
Spirals2	50.08%	*64.88%*	*59.59%*	58.92%	± 0.0392
Big K-means	Min	Max	Median	Mean	SD
Cassini	*65.42%*	99.28%	65.47%	66.14%	± 0.0478
Cuboids	*71.55%*	**100.0%**	*87.24%*	81.19%	± 0.0916
Hypercube	62.55%	**100.0%**	*81.56%*	*83.65%*	± 0.1212
Shapes	*62.60%*	**100.0%**	**100.0%**	*87.76%*	± 0.1726
Simplex	34.18%	**100.0%**	**100.0%**	94.24%	± 0.1492
Smiley	43.37%	94.60%	63.49%	73.83%	± 0.1820
Spirals1	50.00%	50.00%	50.00%	50.00%	± 0.0000
Spirals2	**59.24%**	55.37%	59.24%	*59.29%*	± 0.0006

Table 5.3: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the synthetic datasets using 50.000 instances. Values in bold shows the best results while italics shows the second.

perfectly. However, SC has more problems to discriminate the cluster distribution. It might be because the algorithm has to deal with large data quantities and that introduces noise during the spectrum calculation. When the analysis is focused on the 50.000 instances, MOGGSC also

Comparison	Obs. Dif.	Crit. Dif	Differente
K-means–SC	4	9.3239	FALSE
K-means–MOGGSC	9.5	9.3239	TRUE
SC–MOGGSC	5	9.3239	FALSE

Table 5.4: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets of 50000 instances.

obtains a perfect classification, while SC+N has problems to discriminate the data and it is less stable than the rest of the algorithms according to its standard deviation.

In the case of **Shapes**, the best results according to the Mean and Median are achieved by EM, K-means and MOGGSC (100.0% for K-means and MOGGSC in Median and 99.9% for EM in Mean). Besides, SC obtains the worst results again. According to the results of the 50.000 instances datasets, MOGGSC and K-means obtain again the best results according to the Median, and MOGGSC according to the Mean (98.59%).

Simplex is easy for all the algorithms. The Median value shows that they obtain the maximum results. However, according to the Mean, only MOGGSC and EM keeps these results in all the iterations. The large datasets show similar results, all the algorithms achieve the maximum value according to the Median and only MOGGSC keeps the maximum value according to the Mean.

Smiley results show that the dataset can not only be discriminated using continuity-based approaches but also with Gaussian distributions. These results can be deduced according to the EM (99.88% Median and Mean), MOGGSC (100.0% of Median and 99.31% of Mean) and SC (98.01% Median) results. K-means obtains the worst results due to the shape-based nature of the dataset. Analysing the large datasets, MOGGSC keeps its results but SC+N obtains worse results due to its instability.

The **Spiral1** dataset tests how the algorithms can deal with continuity datasets without noise. In this case, Spectral obtains the best results (100.0%) followed by MOGGSC. Due to K-means and EM are parametric algorithms, they are not able to obtain better results. The large datasets analysis shows similar results, SC+N obtains the best accuracy values while MOGGSC obtains similar values.

The **Spiral2** dataset introduces noise to the previous one. In this case, the results are similar than in the Spiral1 case, however, MOGGSC obtains worse results for the 10.000 instances datasets. This should be a consequence of the data structure which is not clearly discriminated during the micro search. SC obtains the best results (99.80% of Mean and Median) and the rest of the algorithms are not able to discriminate the spirals due to their design. For the large datasets, instead, MOGGSC obtains better results than SC+N. This might be because when the number of instances is increased, the noise also increases and SC+N is not able to deal with it.

The Friedman test p-value for the 10000 instances experiment is 0.03054 and for 50000 instances experiment is 0.0302, which means that there is significantly difference among the distribution. Table 5.2 and 5.4 shows that K-means and Big K-means are statistically different to MOGGSC, and also the original Spectral Clustering. According to the Wilcoxon test, MOGGSC obtains better values than SC+N in four datasets (see Table 5.3).

To conclude, MOGGSC shows competitive results when it is applied to synthetic and continuity-based data, however, it has some problems related to the noise effect during the clustering discrimination process.

5.4.2 Real-World experiments

This section shows the experimental results applied to real-world datasets. Table 5.5 summarizes the results of these experiments.

MOGGSC	Min	Max	Median	Mean	SD
Covtype	15.14%	*39.05%*	**30.72%**	**30.10%**	± 0.0814
Electricity	**58.42%**	**62.31%**	**61.25%**	**60.23%**	± 0.0422
Statlib	6.842%	9.420%	8.244%	8.121%	± 0.0126
Block	**84.41%**	**93.31%**	**▲86.91%**	**87.18%**	± 0.0443
Big K-means	Min	Max	Median	Mean	SD
Covtype	**21.60%**	24.37%	23.28%	23.22%	± 0.0066
Electricity	52.20%	54.30%	52.20%	52.70%	± 0.0090
Statlib	*9.687%*	**10.41%**	**9.992%**	**10.01%**	± 0.0014
Block	51.16%	51.17%	51.16%	51.16%	± 0.0001
SC+N	Min	Max	Median	Mean	SD
Covtype	*18.28%*	**39.82%**	*29.34%*	*29.00%*	± 0.0683
Electricity	*57.65%*	*60.22%*	*60.02%*	*59.48%*	± 0.1223
Statlib	**9.819%**	*9.993%*	*9.947%*	*9.920%*	± 0.0009
Block	*76.61%*	*77.27%*	*76.94%*	*76.94%*	± 0.0047

Table 5.5: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the offline algorithms to the real-world datasets. Values in bold shows the best results while italics shows the second.

Covtype dataset is difficult to discriminate by all the algorithms. MOGGSC and SC+N obtain the best results (30.72% of Median, 30.10% of Mean for MOGGSC and 29.34% of Median, 29.00% of Mean for SC+N). This means that these algorithms are able to deal with massive data in highest dimensions while Big K-means has more problems to discriminate the data, probably generated by the high number of dimensions.

Electricity dataset is the smallest dataset of the whole group. The results show that MOGGSC is able to generate the best discrimination of the data compared against Big K-means and SC+N, however, considering the number of classes, this discrimination (61.25% of Median and 60.23% of Mean) is not representative.

In the case of **Statlib** dataset, Big K-means and SC+N obtain the best values (9.992% of Mean, 10.01% of Median for Big K-means and 9.947% of Median, 9.920% of Mean for SC+N). MOGGSC obtains close results, however, considering that the number of classes is high, these algorithms are not a good choice to identify them.

Block results show that MOGGSC is a good choice to deal with massive data with a small

number of classes and dimension. In this case, the algorithm achives good results with a high different considering the rest of the algorithms (86.91% of Median and 87.18% of Mean).

The Friedman test p-value is 0.3679, which means that there is no statistical significance between the algorithms. However, the Wilcoxon test shows that the algorithm improve the results of the benchmark SC+N algorithm in one dataset (see Table 5.5).

5.5 Dealing with Large and Stream Data Online

Previous sections were focus on dealing with the stream clustering problem offline. However, it is also frequent to deal with this problem in an online mode. In order to adapt the algorithm for online clustering, it is necessary to center the problem on the search space. In this case, the centroid position is changing every time a new data instance is coming in real-time. The algorithm is connected to the data stream and it is modified by every instance. During this work, we have designed an algorithm which deals with this issue, in this case, we have used the online version of K-means for the micro-search combined with MOGGC for the macro-search.

5.6 The Multi-Objective Genetic Graph-based Online Clustering Algorithm (MOGGOC)

The Multi-Objective Genetic Graph-based Online Clustering Algorithm have been designed focused on an online clustering algorithm architecture (see Figure 5.4). This architecture is divided in the following parts:

- **The data stream.** This part controls the data flow, each time the algorithm is ready to read a new instance, the data flow generates that instance. This stream can be connected to different sources such as web-pages, sensors, or even data files.
- **The selection-update (micro-search)**. This module chooses the closest centroid to the new data instance and updates its position. It uses the Online K-means algorithm (see Section 2.7.3). The update process is performed using the following criteria:

  $$c_{q^*}^{(new)} = c_{q^*} - \zeta(x_i - c_{q^*})$$

  where c_{q^*} is the closest centroid, x_i is the data instance and $\zeta = \frac{1}{N_i}$. N_i is the number of instances which has been read until that moment.
- **The manifold discrimination (macro-search)**. Using previous information about the centroid position, the MOGGC algorithm (see Section 4.1) is able to calculate the manifolds. This process can be performed any time because there is not feedback between the two steps.

With these two processes, the algorithm is able to discriminate the clusters in two main levels. Next section describes how it has been evaluated using different datasets.

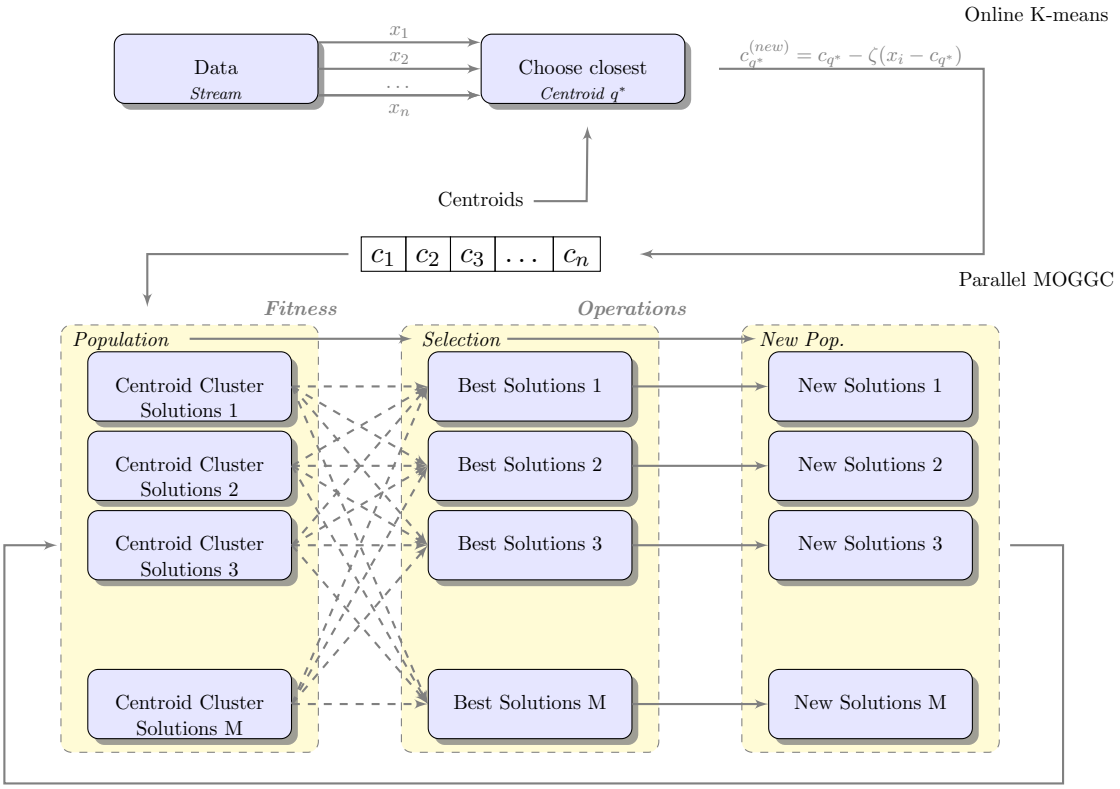


Figure 5.4: MOGGOC algorithm. The micro-search is an Online version of K-means while the macro-search is the parallel version of MOGGC.

5.7 MOGGOC validation

Actually, the evaluation process to measure the quality of an online algorithm is an open-problem. In this case, we have compared the algorithm using the previous evaluation metrics and online algorithms.

5.7.1 Algorithm complexity: Time and Memory

The complexity of the algorithm depends on the number of instances and also the number of centroids. The complexity of MOGGC step is the same that MOGGSC (see Section 5.3.1). However, the online clustering part is the same than online k-means. It is, $O(N \cdot log(N))$, where N is the number of data instances.

The total amount of the whole process is

$$O(N \cdot log(N) + G \cdot (c^2 + (P + \bar{P})^2 \cdot \log(P + \bar{P}))). \tag{5.6}$$

where G is the number of Generations, P and $\bar{P}$ are the populations of the SPEA2 algorithms and c is the number of centroids. The first part of the equation is the Online K-means and the second is MOGGC.

The memory consumption of MOGGOC can be calculated as:

$$c \cdot K, \tag{5.7}$$

due to the algorithm does not keep information about the instances. Comparing this with MOGGSC and MOGGC, we have to consider that these algorithms depend on the number of instances N which is bigger than c ($N >> c$). K is the neighbourhood of the Similarity Graph which is a constant value. Hence, MOGGOC consumption is significantly smaller than MOGGC ($N \cdot K$) and MOGGSC ($N + c \cdot p + c \cdot K$).

5.7.2 Experimental Setup

MOGGOC has been compared against three online clustering algorithms: CluStream, Online K-means and ClusTree (see Section 2.7.2). All these algorithms have been set to generate a double search (micro and macro). The macro search which has been applied uses the KNN algorithm.

It is also important to describe how the data streams works. In this case, we have used the data streams structures from the R package called stream[4]. In our approach, we read the data stream randomly in order to measure the quality of the algorithm identifying the clusters.

The metric used for all the algorithms is the Euclidean Distance. The genetic parameters, for all the experiments, have been set as: 100 generations, 300 population, $(\mu + \lambda)$ selection, 5 elitism, 0.1 mutation, 0.5 crossover. The similarity metric used is RBF.

[4]http://cran.r-project.org/web/packages/stream/index.html

5.8 Experimental Results

This section shows the experimental results of the online clustering algorithm. Following the same methodology that was used in Section 5.4, we have compared the algorithm with the datasets described in Section 5.3.3, in order to measure and compare the quality of the algorithm in synthetic and real-world domains.

5.8.1 Experiments with Synthetic data

Table 5.6 shows the results for the algorithms applied to the synthetic datasets using 50.000 instances. As we can appreciate, MOGGOC obtains generally the best results but it is important to analyse the algorithm according to each dataset in order to identify its weaknesses.

In the case of **Cassini** dataset, MOGGOC obtains the best results of Median and Mean, followed by Clustree which also obtains good Mean values (74.31%).The dataset (see Figure 5.3 (a)), as was mentioned before, has two sections which can be easily discriminated by a Gaussian estimator. However, for online algorithms these areas might suppose an important variation in the centroid trends.

For **Cuboids** dataset, MOGGOC also achieves the best results followed by Clustream which obtains the same results for the Median and the Maximum but lower results for the Mean (89.89% of accuracy). Online K-means and Clustree also obtain good maximum results, but their accuracy according to the Median (88.84% and 72.74%, respectively) and the Mean (84.12% and 81.21%) is lower. This should be because these algorithms are influenced by the middle cube (see Figure 5.3) which modifies the centroid trends for these algorithms.

Hypercube results show that the algorithms can not discriminate perfectly spherical clusters. MOGGOC obtains the best results again (100.0% of Median and 99.69% of Mean) followed by Clustream (also 100.0% of Median and 98.22% of Mean). This shows that these algorithms are less influenced by the noise effects produced during the stream reading. However, Clustree and Online K-means are more sensitive to these effects even when they need to discriminate simple clusters such as spheres.

In the case of **Shapes**, all the algorithms achieve the maximum results for the Max and the Median. However, only MOGGOC and Clustream keep this value in the Mean. This suggests that Online K-means (93.57% of Mean) and Clustree (84.59%) have also small problems with the different trends that are generated by the shapes.

Simplex is easy for all of the algorithms considered. The Median value shows that they obtain the maximum results, again. However, according to the Mean, only MOGGOC and Clustream keep these results in all the iterations. It is important to remark that Clustree and Online K-means have similar problems with this dataset as they had with Cuboids and Hypercube, this fact suggests that these algorithms are less stable for volumetric data.

Smiley shows that the online are more sensitive during the discrimination of shape-based data, even when these shapes might be discriminated using Gaussian models. MOGGOC presents the best results (100.0% of Median and 98.74% of Mean), followed by Clustream (91.60%

MOGGOC	Min	Max	Median	Mean	SD
Cassini	60.70%	**98.55%**	**75.52%**	**83.13%**	± 0.1531
Cuboids	**85.20%**	**100.0%**	**100.0%**	**99.41%**	± 0.0292
Hypercube	**84.73%**	**100.0%**	**100.0%**	**99.69%**	± 0.0216
Shapes	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	**68.54%**	**100.0%**	**100.0%**	**98.74%**	± 0.0623
Spirals1	**50.05%**	**80.05%**	**57.12%**	**57.67%**	± 0.0565
Spirals2	**55.62%**	64.10%	**62.52%**	**62.71%**	± 0.0198
Online K-means	Min	Max	Median	Mean	SD
Cassini	*65.42%*	65.52%	65.47%	65.46%	± 0.0003
Cuboids	71.65%	**100.0%**	88.84%	84.12%	± 0.1219
Hypercube	62.66%	**100.0%**	81.36%	83.75%	± 0.1122
Shapes	62.63%	**100.0%**	**100.0%**	93.57%	± 0.1389
Simplex	*62.53%*	**100.0%**	**100.0%**	92.53%	± 0.1508
Smiley	42.80%	94.11%	62.54%	78.99%	± 0.1729
Spirals1	50.00%	50.00%	50.00%	50.00%	± 0.0000
Spirals2	**59.36%**	*59.36%*	59.36%	*59.36%*	± 0.0000
Clustree	Min	Max	Median	Mean	SD
Cassini	**65.75%**	*96.75%*	66.97%	*74.31%*	± 0.1301
Cuboids	71.46%	**100.0%**	72.74%	81.21%	± 0.1108
Hypercube	62.73%	**100.0%**	82.29%	86.75%	± 0.1126
Shapes	62.54%	**100.0%**	**100.0%**	84.59%	± 0.1765
Simplex	62.52%	**100.0%**	**100.0%**	88.96%	± 0.1703
Smiley	42.15%	95.38%	64.41%	74.64%	± 0.1608
Spirals1	*50.00%*	50.08%	*50.01%*	*50.03%*	± 0.0002
Spirals2	50.19%	*64.32%*	58.84%	58.64%	± 0.0295
Clustream	Min	Max	Median	Mean	SD
Cassini	64.41%	79.91%	67.39%	68.89%	± 0.0478
Cuboids	*71.67%*	**100.0%**	**100.0%**	*89.89%*	± 0.1157
Hypercube	*81.43%*	**100.0%**	**100.0%**	*98.22%*	± 0.0541
Shapes	**99.96%**	**100.0%**	**100.0%**	**100.0%**	± 0.0001
Simplex	**100.0%**	**100.0%**	**100.0%**	**100.0%**	± 0.0000
Smiley	*62.14%*	*95.73%*	*91.60%*	*80.79%*	± 0.1451
Spirals1	*50.00%*	*50.38%*	*50.01%*	*50.03%*	± 0.0006
Spirals2	50.88%	**64.88%**	*59.59%*	58.92%	± 0.0392

Table 5.6: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the synthetic datasets. Values in bold shows the best results while italics shows the second.

of Median and 80.79%). The rest of the online algorithms obtain values around the 65% of Median and 80% of Mean which means that they are really sensitive to the centroid trends produced by the shapes.

Comparison	Obs. Dif.	Crit. Dif	Differente
On. K-means–Clustree	2.5	11.59	FALSE
On. K-means–Clustream	10.5	11.59	FALSE
On. K-means–MOGGOC	15.0	11.59	TRUE
Clustree–Clustream	8	11.59	FALSE
Clustree–MOGGOC	12.5	11.59	TRUE
Clustream–MOGGOC	4.5	11.59	FALSE

Table 5.7: Multiple Comparison of Friedman Test for the algorithms applied to the synthetic datasets.

The **Spiral1** dataset shows that only MOGGOC is able to provide a competitive result compared against offline algorithms. However, these results are not significantly good, the structure of the data in the space produces that the centroid trends are constantly changing and making the Voronoi regions less significant. It supposes that these regions can not be easily joined in the macro level. The rest of the algorithms probably set the centroids in intermediate positions of the spirals (see Figure 5.3 (g)).

The **Spiral2** dataset introduces noise to the previous one, as was explained before. In this case, the results are unstable. However, MOGGOC obtains the best results (62.52% of Median and 62.71% of Mean) compared with the rest of the algorithms. The results of these algorithms should also be a consequence of the centroid position in intermediate positions.

The Friedman test p-value is 0.001566. This means that there is statistical difference among the distributions. Table 5.7 shows the main differences among the algorithms: Online K-means and Clustree are statistically different to MOGGOC.

To conclude, MOGGOC shows good accuracy results as an online algorithm, however, there are some issues such as the adaptation to noisy datasets, or curve deformations, which need to be studied.

5.8.2 Real-World experiments

Once the MOGGOC algorithm has been evaluated on synthetic environments, it is also important to evaluate it in real environments in order to measures its performance. Table 5.8 shows the results of the application of the online algorithm to the datasets described in Section 5.3.3.

Covtype results show that MOGGOC obtains good results compared with the rest of the algorithms (29.85% of Median and 28.80% of Mean). This means that the algorithm is able to deal with massive data in higher dimensions, while the rest have more problems to discriminate the data, probably generated by the high number of dimensions.

The case of **Electricity** is different. The results show that MOGGOC is able to generate a good online discrimination according to the Median (54.16%). However, its Mean is lower than Online K-means and Clustree. Again, considering the number of classes, this discrimination is not representative.

In the case of **Statlib** dataset, Online K-means and Clustree obtain the best values (9.974% of Mean, 9.985% of Median for Online K-means and 9.756% of Median, 9.690% of Mean for

MOGGOC	Min	Max	Median	Mean	SD
Covtype	*16.35%*	**39.05%**	**29.85%**	**28.80%**	± 0.0672
Electricity	50.53%	54.16%	**54.16%**	52.65%	± 0.0157
Statlib	4.944%	5.390%	5.268%	5.225%	± 0.0014
Block	**86.91%**	**92.31%**	**86.91%**	**87.18%**	± 0.0121
Online K-means	Min	Max	Median	Mean	SD
Covtype	**17.01%**	23.29%	23.26%	22.94%	± 0.0140
Electricity	**52.20%**	**54.30%**	*52.20%*	*52.72%*	± 0.0093
Statlib	**9.705%**	*10.20%*	**9.985%**	**9.974%**	± 0.0014
Block	*50.29%*	52.45%	51.42%	51.36%	± 0.0062
Clustree	Min	Max	Median	Mean	SD
Covtype	15.86%	25.59%	19.17%	20.26%	± 0.0319
Electricity	**52.20%**	**54.30%**	*52.20%*	**52.83%**	± 0.0099
Statlib	*8.983%*	**10.45%**	*9.756%*	*9.690%*	± 0.0040
Block	50.11%	59.21%	55.22%	54.59%	± 0.0283
Clustream	Min	Max	Median	Mean	SD
Covtype	15.73%	29.51%	19.03%	20.03%	± 0.0422
Electricity	**52.20%**	52.20%	*52.20%*	52.20%	± 0.0000
Statlib	6.913%	10.04%	8.276%	8.365%	± 0.0084
Block	50.24%	*63.13%*	*55.94%*	*56.43%*	± 0.0342

Table 5.8: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the real-world datasets. Values in bold shows the best results while italics shows the second.

Clustree). MOGGOC obtains the worst results, this means that the algorithm has problems to deal with a high number of classes. This problem is probably generated during the macro-search.

Block results show that MOGGOC is a good choice to deal with massive data with a small number of classes and dimension. In this case, the algorithm achieves good results considering the rest of the algorithms (86.91% of Median and 87.18% of Mean).

Finally, the Friedman test p-value is 0.5062 which says that there is not significantly difference among the algorithms.

5.9 Final Discussions

During this section two new algorithms for stream data analysis have been developed. These algorithms have shown competitive results compared with the algorithms extracted from the literature. However, there are some issues that need to be deeply study about this new online algorithm:

- It is interesting to study how adapt the number of centroids in the micro search.

- It is important to analyse the influence of data order, due to some clusters might disappear and the trends might also be part of the clustering discrimination.
- It is also important to find a way to generate some feedbacks between the macro-search and the micro-search.

These ideas might be used for future extensions in order to consider them as a starting point for a new algorithm generation.

Chapter 6

Other Bio-inspired Approaches: ACO Clustering

"No matter how far a person can go
the horizon is still way beyond you."

- Zora Neale Hurston

Unsupervised data mining techniques compose a complex field, where several different approaches have been tested in order to obtain similar or even better results to supervised techniques. The main difference between these two techniques is that supervised techniques use some kind of label (target) information, which is used during the model generation, providing a more accurate model—the accuracy of the model is determined by comparing the prediction with the label information. Unsupervised techniques, instead, are totally blind respect to the label information. An advantage of unsupervised techniques is that they can deal with a huge quantity of (unlabelled) data without a feedback of their performance.

Unsupervised techniques have been studied from different perspectives. Over the last few years, bio-inspired techniques have been widely used, usually based on evolutionary algorithms or swarm intelligence that mimic a natural behaviour —e.g., the evolutionary process in genetic algorithm, collective behaviour in ant colony optimization. This chapter has been focused on the latter, which is becoming a promising field for unsupervised techniques. ACO algorithms are based on the foraging behaviour of ant colonies when they try to find the optimal path between their nest and a food source. Based on this idea, researchers have created several optimization algorithms in data mining, which have been focused on the path optimization process followed by the ants to create solutions for hard optimization problems [129, 157, 158].

The work presented in this chapter is focused on the application of ACO in the unsupervised learning task of clustering, where the goal is to group similar data points in the same cluster and, at the same time, to maximise the difference between different clusters. The first part has been focused on a first approach to the problem through medoid-based algorithms while the second part is focused on continuity-based clustering. In the same way that GGC and MOGGC, it is inspired by the Spectral Clustering (SC) algorithm [155]. Both approaches use the ACO-based Clustering algorithm (ACOC), proposed by Kao and Cheng [100] as a starting point. ACOC is a centroid-based clustering algorithm, which tries to optimize the centroid (central point) position of each cluster. Following this idea, we have focused the proposed algorithms on addressing a medoid and a spectral-based approach. Inspired by other clustering algorithms [139, 24], we have

reformulated the original ACOC algorithm to create the new algorithms. In order to check the performance of the proposed algorithms, we have compared them against well-known clustering algorithms using synthetic and real-world datasets that have been used in previous chapters.

6.1 Studying the Medoid-based approach

Analysing the possibilities of ACOC algorithm, we present a new version of the centroid-based ACOC algorithm. The following sections are focused on addressing the centroid-based approaches problems: they need to know the properties of the search space in order to determine the central point, and they are sensitive to noise effects. Inspired by other clustering algorithms [139, 24], we reformulated the original ACOC algorithm in a different way to create a medoid-based algorithm. Medoid-based clustering algorithms are usually more robust to noise effects, and do not need the properties of the search space to find a solution—they usually have the distance amongst the data instances, which can be obtained as a Gram matrix of a kernel or a distance measure, and they try to choose those data instances to define the best clusters. These selected instances are called medoids.

In order to check the performance of the proposed algorithm, we have compared it against the original ACOC algorithm using synthetic and real-world datasets and also included the well-known clustering algorithms PAM (Partition Around Medoids) [102] and K-means [126] in the comparisons.

6.2 Medoid-based ACO Clustering Algorithm (MACOC)

This section presents the Medoid-based ACO Clustering Algorithm (MACOC). The MACOC algorithm is similar to the Partition Around Medoids (PAM) algorithm, where the goal of the algorithm is to choose the best M medoids (data instances) based only on distance information. This kind of algorithms usually use a dissimilarity/similarity matrix that measures the distances between the data points. The medoid-based approach is a generalization of the centroid-based approach, but in the medoid case, the properties of the search space are not required—only the distances between the data points.

As an ACO algorithm, MACOC algorithm is based on the ACOC algorithm [100]. They have a similar search graph, where the ants try to define the optimal cluster assignment for each of the instances (data points). This graph is based on instances and clusters (Figure 6.1). It has an associated NxM matrix, where N is the number of instances and M is the number of clusters (medoids). While in the case of ACOC the construction graph is full-connected, MACOC uses a graph divided in levels, where each instance defines a level (Figure 6.1). The nodes are visited following the sequence of instances, therefore the graph is not full-connected, reducing significantly the memory usage and the complexity of the solution space. It should be noted that this does not incorporate any bias in the search, since the order of the instances is not relevant.

The algorithm is based on several ants looking for the best path in the construction graph. Each ant (k) has the following features:

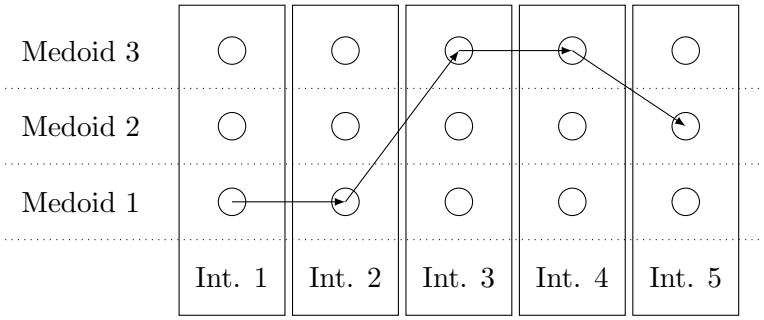


Figure 6.1: Representation of the ant travelling around the search graph. Ant visit each instance in order to assign them to a medoid based on the heuristic information and pheromone levels.

- Set of chosen medoids M^k (which are randomly selected).
- Weight matrix W^k (based on the distance between the instances and the ant's medoids).

The ant has two possible search strategies, exploration and exploitation, similar to the ACOC algorithm. It chooses the strategy for the cluster assignation j according to the following formula:

$$j = \begin{cases} argmax_{u \in N_i}\{[\tau(i,j)][\eta(i,j)]^\beta\} & \text{, if } q \leq q_0 \\ S & \text{, otherwise} \end{cases} , \tag{6.1}$$

where N_i is the set of nodes associated to instance i (see Figure 6.1), j is the chosen cluster, $\tau(i,j)$ is the pheromone value between i and j, q_0 is the user-defined exploitation probability, q is a random number for strategy selection, $\eta(i,j)$ is the heuristic information between i and j, and S is the ACO-based search strategy. The heuristic information between an instance i and a candidate medoid j is defined by the formula:

$$\eta(i,j) = \frac{1}{d(i,j)} , \tag{6.2}$$

and the ACO-based exploration strategy S defined by:

$$S = P(i,j) = \frac{[\tau(i,j)] \cdot [\eta(i,j)]^\beta}{\sum_{l=1}^{m} [\tau(i,l)] \cdot [\eta(i,l)]^\beta} . \tag{6.3}$$

One of the main differences between ACOC and MACOC is that MACOC keeps more information about the ants movements in the pheromone matrix. In the case of ACOC, the pheromone matrix is a relationship between the instance and the centroid-label (i.e., the index of the cluster), which is not the centroid itself. In the case of MACOC, the pheromone matrix is a relationship between the instance and the medoid (another data instance), which means that if/when the medoid-label changes as a result of the random selection process, the previous pheromone value is still available. In other words, in the ACOC algorithm, if the centroid value that was previously used as the centroid c_1 is used as the centroid c_2, the previous pheromone values are lost, since they are associated with the label (position) c_1; in the MACOC algorithm,

if the medoid instance that was previously used as medoid m_1 is used as the medoid m_2, the previous pheromone values are still used, since the pheromone is associated with the data instance and not with the medoid label.

The MACOC algorithm can be described as follows:

1. Initialize the pheromone matrix (τ_0).
2. Initialize the ants: choose n random medoids for M^k (where n represents the number of clusters) and set the matrix W^k to 0.
3. For each ant:
   (a) Select the next data object i.
   (b) Select a cluster j:
      i. Choose a strategy
      ii. Calculate neighbouring nodes probability.
      iii. Visit the node.
   (c) Update W^k.
4. Choose the best solution:
   (a) Calculate the objective function for each ant:

$$J^k = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij}^k \cdot d(x_i, m_j^k) \ , \tag{6.4}$$

   where $w_{ij}^k \in W^k$ and d is a distance function.
   (b) Rank the ants solutions.
   (c) Choose the best ant (iteration-best solution).
   (d) Compare it with the best-so-far solution and update this value with the maximum between them.
5. Update the pheromone trails (global updating rule): only the r best ants are able to add pheromones:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{h=1}^{r} w_{ij}^h \cdot \Delta\tau_{ij}^h \ , \tag{6.5}$$

   where ρ is the pheromone evaporation rate, $(0 < \rho < 1)$, $w_{ij}^h \in W^h$, t the iteration number, r is the number of elitism ants and $\Delta\tau_{ij}^h = 1/J^k$ is the quality of the solution created by ant h.
6. Check termination condition:
   (a) If the number of iterations is greater than the total iterations: re-centralise the instances assigning each data point to its closest medoid and finish.
   (b) Otherwise, go to step 2.

6.3 Experiments

This section presents the experiments which have been carried out to measure the quality of the proposed MACOC algorithm. The comparisons have been carried out against K-means, PAM and ACOC algorithms.

6.3.1 Datasets Description

For the synthetic experiments we have created the following datasets:

- *Synthetic Data 1*: This dataset is formed by 9 two-dimensional Gaussian models and in this case, there are 3 gaussians which are closer than the rest.
- *Synthetic Data 2*: This second dataset is also formed by 9 two-dimensional Gaussian models, however, in this case, there are noisy data in the background.

For the real-world experiments, we have chosen four datasets extracted from UCI Machine Learning Repository [70]:

- *Iris*: Contains 50 instances distributed over 3 classes, with 4 attributes each.
- *Wine*: Contains 178 instances distributed over 3 classes, with 13 attributes each.
- *Vertebral Column* (Ver. Col.): Contains 310 instances distributed over 3 classes, with 6 attributes each.
- *Breast Tissue* (Bre. Tis.): Contains 106 instances distributes over 6 classes, with 10 attributes each.

6.3.2 Experimental Setup and Evaluation Methods

This section describes the selected algorithms used to measure the MACOC quality.

K-means [126] is an iterative algorithm based on centroids, which are randomly selected at the beginning. The goal of the algorithm is to find the best centroid positions. It is executed in two steps: in the first step, it assigns the data to the closest centroid (cluster); and in the second, it calculates the new position of the centroid as a centroid of the data which has been assigned to it.

PAM [102] is similar to K-means, but it used medoids instead of centroids. PAM can works with a dissimilarity/similarity matrix, which is used to calculate the cost of each medoid belonging to a cluster.

ACOC [100] is the baseline algorithm used to create MACOC. It works with centroids and ants. The main difference, apart of the algorithm centroid nature, is that ACOC uses a pheromone matrix from the data instances to the centroid-labels, while our algorithm use

a pheromone matrix between all the data to remember the previous medoid assignation. The parameters of ACOC and MACOC algorithms have been set in a similar way to the original work [100]: the number on ants is 10, the number of elitism is 1, the exploration probability is 0.0001, the initial pheromone values follow an uniform distribution $[0.7, 0.8]$, $\beta = 2.0$, $\rho = 0.1$, and the maximum number of iterations is 1000. The only difference is that the MACOC initial pheromone values have been set as $\frac{1}{n}$ (where n is the number of clusters).

All the experiments have been carried out using the Euclidean distance as the metric, defined by:

$$d(x_i, x_j) = ||x_i - x_j|| = \sqrt{\sum_q (x_i^q - x_j^q)^2} \ , \tag{6.6}$$

where x_i, x_j represent two data instances and q represents each attribute of the data instance. Additionally, all algorithms need the number of cluster as an initial parameter.

The evaluation of the experiments has been focused on two different ideas: the synthetic dataset has been evaluated according to the cluster discrimination and the performance of the algorithm to discriminate the original clusters in the noisy case; the real-world datasets have been evaluated using the accuracy.

6.3.3 Synthetic Experiments

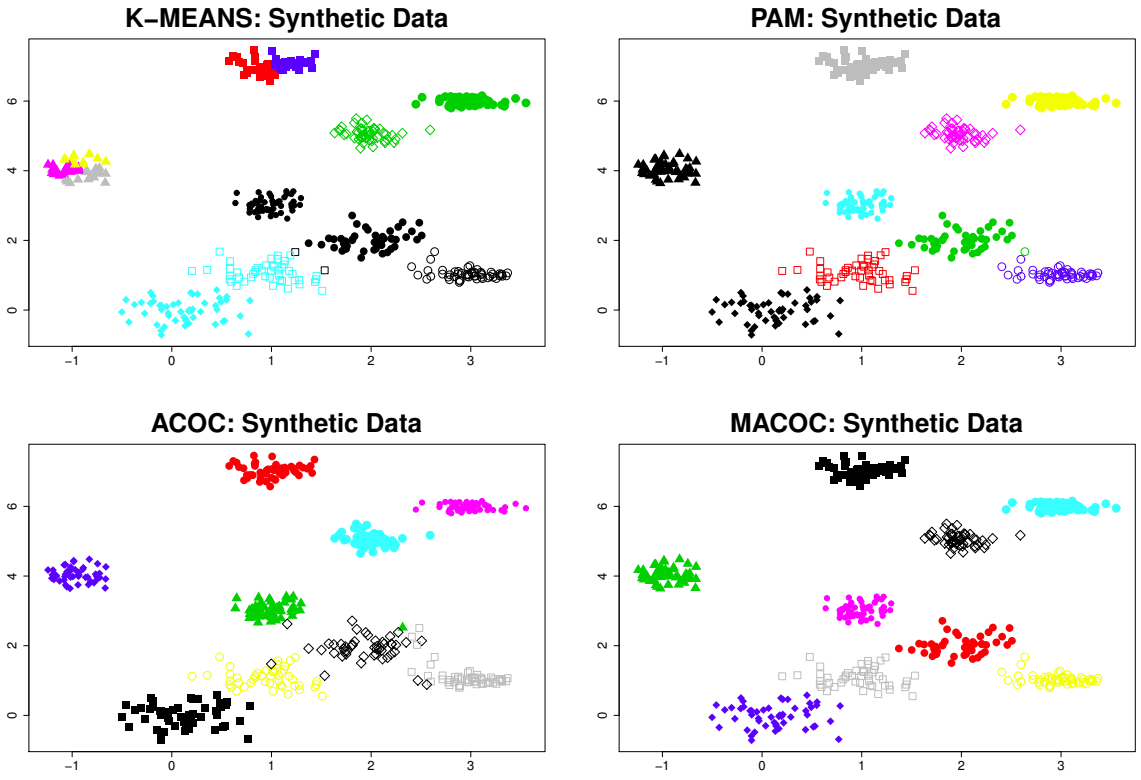


Figure 6.2: Results for 9 gaussian distribution.

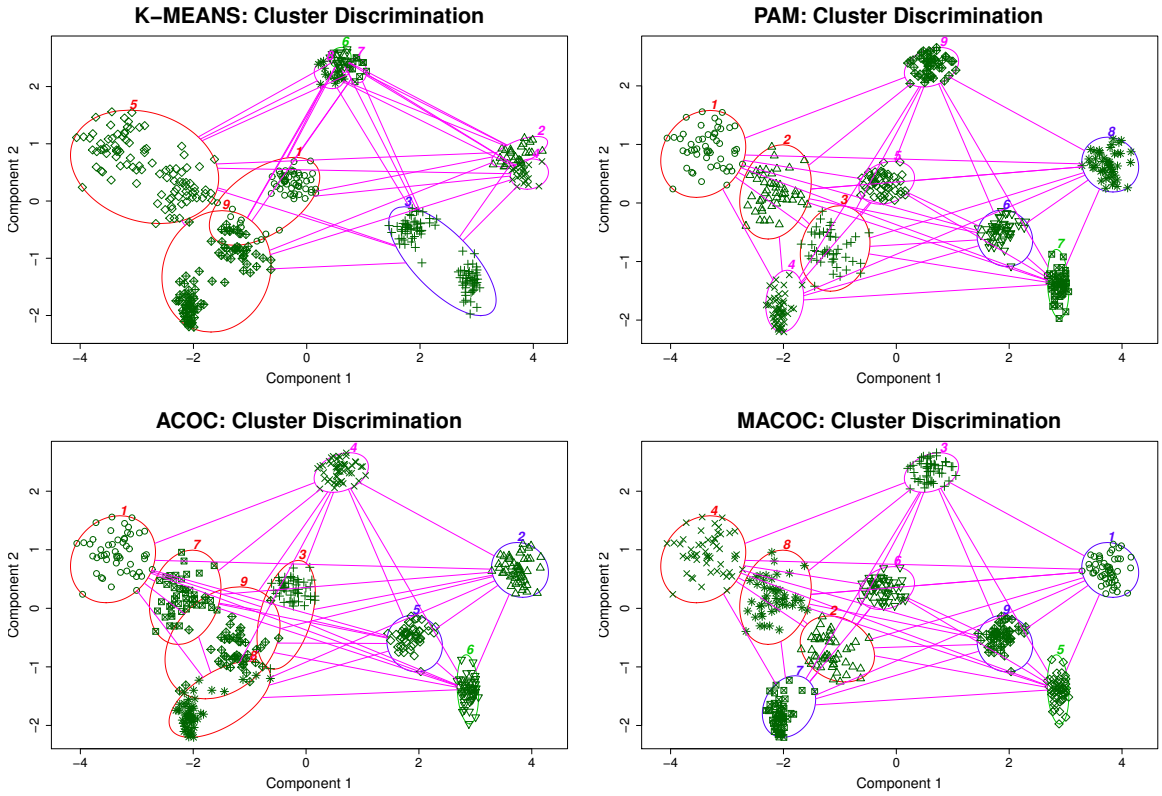


Figure 6.3: Discrimination results for the synthetic 9 gaussian distribution.

The first synthetic dataset is generally easy for all the algorithms (see Figures 6.2, 6.3 and 6.3). The discrimination of the clusters is clearer in this case, resulting in a clear separation of the clusters. The only algorithm that has several problems to identify the clusters is K-means (see Figure 6.3 and Table 6.3) –probably a result of an early convergence to local minimal solution. PAM provides a good solution of the cluster discrimination and also provides a stable solution (its standard deviation is 0). It means that the algorithm is able to find the medoids with no problems. In the case of ACOC, the solution discriminates the cluster kernels. However, the boundaries are not well-defined (see Figure 6.3). MACOC obtains good results for both cluster identification and boundary definition, and also accuracy results (see Table 6.3).

The second dataset introduces noise and the noise significantly modifies the behaviour of the algorithms (see Figures 6.4, 6.5 and Table 6.3). K-means is not able to identify the cluster kernels and it joins several clusters together, generating a cluster with noisy data. ACOC is also able to find the kernels and discriminate them, however, the boundaries are not well-defined and several instances overlap with other clusters. Finally, PAM and MACOC achieve similar results. In the case of PAM, there are some boundary problems in the central clusters while in the case of MACOC the boundaries are clearer, except for one instance (see Figure 6.5, at the right of the MACOC image).

These results suggest the following conclusions regarding the comparison with ACOC: while ACOC has boundary problems, which are increased when there is noisy information, MACOC obtains good results for cluster boundary definition and it is more robust to the presence of noise.

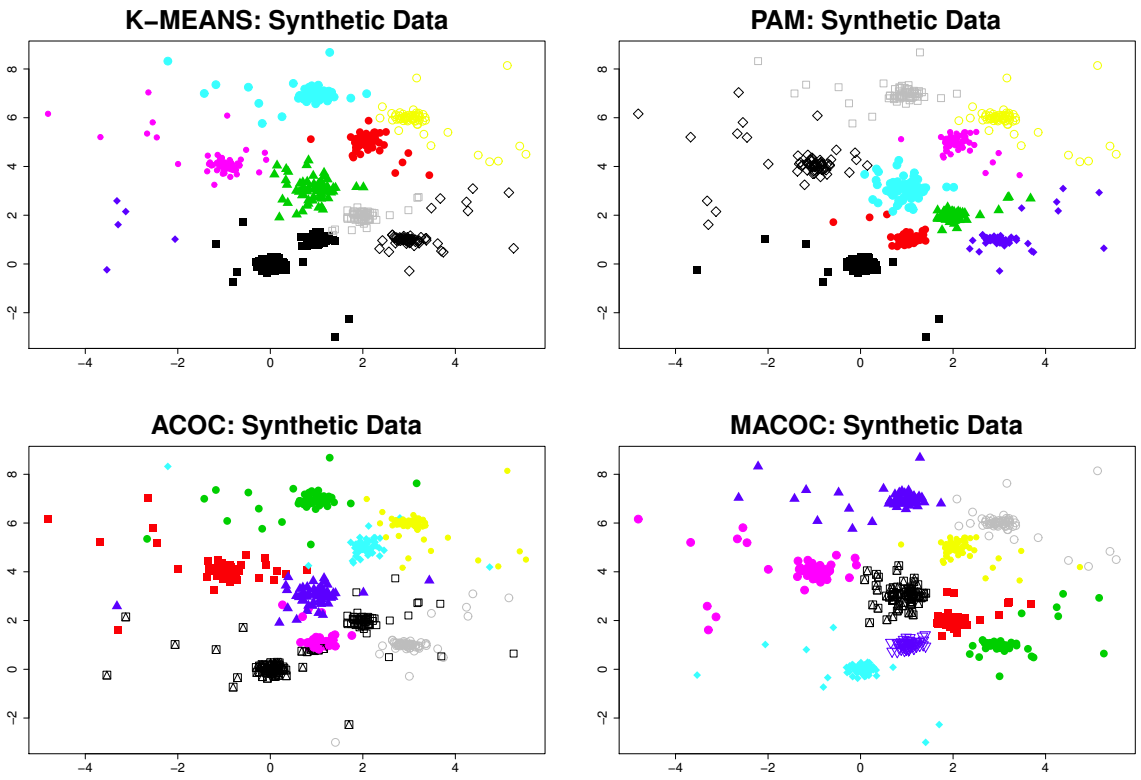


Figure 6.4: Results for 9 noisy gaussian distribution.

6.3.4 Real-World Experiments

Table 6.4 shows the results of the algorithms applied to real-world datasets extracted from UCI Machine Learning repository [70].

In the Iris case, K-means and PAM obtains similar results according to the median. K-means obtains the worst minimum accuracy results (58%) and it is the less robust algorithm (its standard deviation is 0.1311). PAM is the most robust algorithm in this case (0 standard deviation), while ACOC and MACOC obtain similar robustness results. The highest minimum value is achieved by both ACOC and PAM. The highest maximum, mean and median values are achieved by MACOC (95.33%, 90.67% and 90.65%, respectively). While MACOC shows better results than ACOC, these results can not be considered different because the null hypothesis can not be refused according to Wilcoxon Test [201].

The application of the algorithms on Wine dataset has shown that K-means also obtains the worst results according to the accuracy and robustness. PAM obtains the highest minimum value and it is again the most robust algorithm. According to the maximum, mean and median values, MACOC achieves the best results (72.47%, 71.47% and 71.91%, respectively). In this case the null hypothesis is refused with a good significance level, therefore, MACOC results are statistically significantly better than ACOC.

Vertebral column (Ver. Col.) dataset shows different results than the rest of the datasets. In this case, the best algorithm is K-means, which achieves the maximum value for all the metrics.

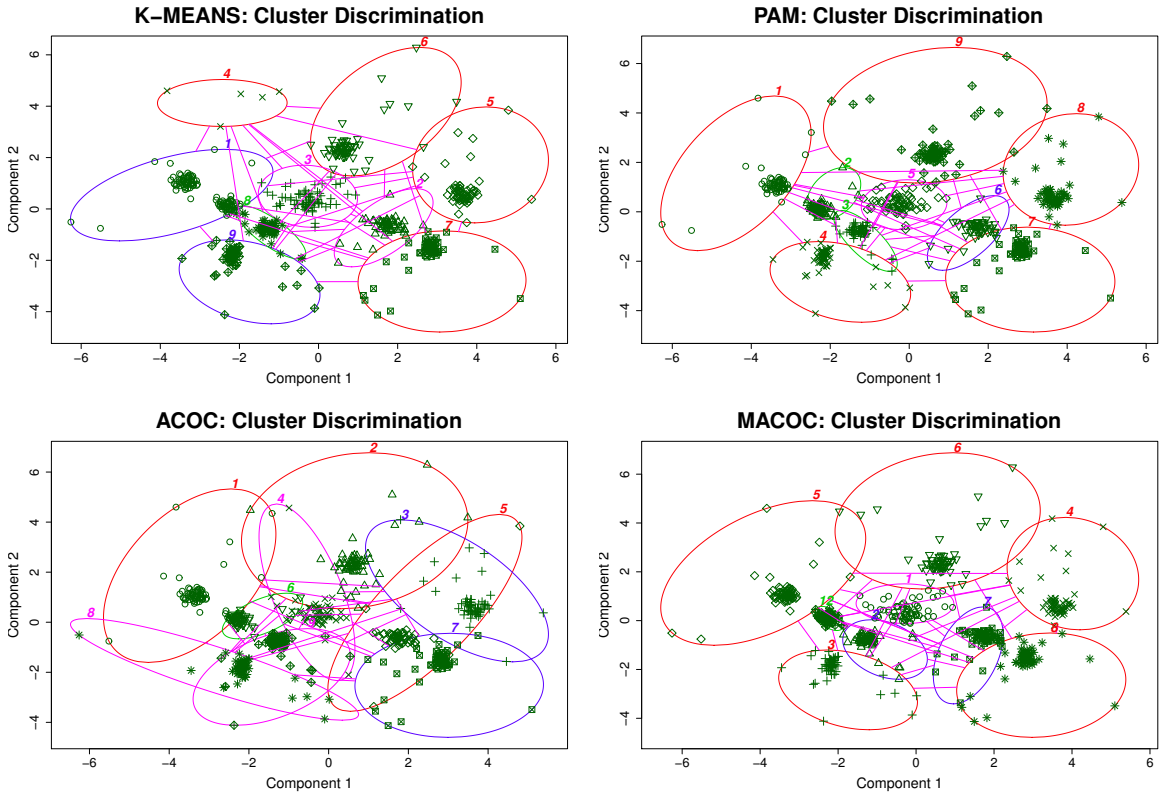


Figure 6.5: Discrimination results for synthetic 9 noisy gaussian distribution.

MACOC is the second algorithm according to median, mean and max (52.58%, 53.26% and 65.48%). Again, the Wilcoxon test shows that the null hypothesis can be refused with a high significance level (3e-05), therefore, MACOC results are statistically significantly better than ACOC.

Finally, Breast Tissue dataset shows that MACOC achieves the best results according to mean, median and max (35.55%, 34.91% and 40.57%). In this case, ACOC and PAM achieves similar results, specially according to the median (33.96 %). The null hypothesis can be refused with a significance level of 0.05 (in this case, Wilcoxon test is 0.023), therefore, MACOC results are statistically significantly better than ACOC.

These results show that MACOC improves the performance over ACOC, given that that the solutions obtained by MACOC and ACOC are usually statistically different according to Wilcoxon test in favour of MACOC. Overall, these results are promising regarding the use of medoids instead of centroid, since this is the main difference between MACOC and ACOC.

6.4 Conclusions of MACOC

MACOC is focused on a medoid-based approach. It is an adaptation from a previously proposed centroid-based ACOC algorithm to a medoid-based approach. From the ACO perspective, the new algorithm has also improved the use of the pheromone, extending the pheromone matrix to

MACOC	Min	Max	Median	Mean	SD
Synthetic 1	99.11%	**100.0**%	**99.78**%	**99.75**%	▲ ± 0.0028
Synthetic 2	96.73%	**100.0**%	**98.18**%	**98.75**%	▲ ± 0.0121
ACOC	Min	Max	Median	Mean	SD
Synthetic 1	92.67%	100.0%	98.89%	98.57%	± 0.0128
Synthetic 2	82.91%	92.27%	95.27%	94.40%	± 0.0314
K-means	Min	Max	Median	Mean	SD
Synthetic 1	56.67%	99.78%	84.00%	80.87%	± 0.1143
Synthetic 2	64.65%	98.00%	76.73%	80.22%	± 0.0741
PAM	Min	Max	Median	Mean	SD
Synthetic 1	**99.78**%	99.78%	99.78%	99.78%	± 0.0000
Synthetic 2	**98.00**%	98.00%	98.00%	98.00%	± 0.0000

Table 6.1: Results of the application of the algorithms to the synthetic datasets. The p-values for the Wilcoxon test applied to ACOC and MACOC results are: Synthetic 1 (2.394e-13) and Synthetic 2 (7.734e-10)—statistical significant improvements are indicated by a ▲ symbol.

keep the information from different candidate medoid instances across iterations.

The application of the algorithm to synthetic and real-world datasets has shown that MACOC is more robust to noisy information and it defines better cluster boundaries than ACOC. It also showed that MACOC has good general results compared with well-known clustering algorithms.

6.5 Spectral-based ACO Clustering Algorithm (SACOC)

This section presents the proposed Spectral-based ACO Clustering Algorithm (SACOC). This algorithm is similar to Spectral Clustering. The goal of the algorithm is to choose the data discrimination representing the information as a similarity graph and cutting it in different clusters.

6.5.1 ACOC algorithm

The ACOC algorithm [100] is the base of SACOC and MACOC. It has a search space based on instances and centroids, and can be defined as a graph whose associated matrix is a N×M matrix, where N is the number of instances and M is the number of centroids (clusters).

The algorithm works with several ants looking for the best path in the graph (see Figure 6.6). Each ant (k) has the following features: a list of visited objects (tb^k), a set of chosen centroids C^k and a Weighted matrix W^k (related to the assignation of objects to clusters).

Any ant k has two possible strategies: exploration and exploitation. It choose the strategy

MACOC	Min	Max	Median	Mean	SD
Iris	87.33%	**95.33**%	**90.65**%	**90.67**%	± 0.0187
Wine	69.10%	**72.47**%	**71.91**%	**71.47**%	▲ ± 0.0075
Ver. Col.	46.13%	**65.48**%	52.58%	53.26%	▲ ± 0.0488
Bre. Tis.	28.30%	**40.57**%	**34.91**%	**35.55**%	▲ ± 0.0328
ACOC	Min	Max	Median	Mean	SD
Iris	**89.33**%	93.33%	90.00%	90.13%	± 0.0080
Wine	70.22%	71.35%	70.79%	70.78%	± 0.0026
Ver. Col.	47.74%	54.19%	49.35%	49.43%	± 0.0095
Bre. Tis.	31.13%	**40.57**%	33.96%	34.47%	± 0.0229
K-means	Min	Max	Median	Mean	SD
Iris	58.00%	89.33%	89.33%	82.44%	± 0.1311
Wine	56.74%	70.22%	70.22%	67.26%	± 0.0564
Ver. Col.	**56.13**%	**65.48**%	**56.13**%	**57.81**%	± 0.0339
Bre. Tis.	33.02%	33.96%	33.02%	33.04%	± 0.0013
PAM	Min	Max	Median	Mean	SD
Iris	**89.33**%	89.33%	89.33%	89.33%	± 0.0000
Wine	**70.79**%	70.79%	70.79%	70.79%	± 0.0000
Ver. Col.	48.71%	48.71%	48.71%	48.71%	± 0.0000
Bre. Tis.	**33.96**%	33.96%	33.96%	33.96%	± 0.0000

Table 6.2: Results of the application of the algorithms to the different datasets extracted from the UCI database. The p-values for the Wilcoxon test applied to ACOC and MACOC solutions are: Iris (0.4439), Wine (4.117e-07), Ver. Col. (3e-05), Bre. Tis. (0.02349)—statistical significant improvements are indicated by a ▲ symbol.

according to the following formula:

$$j = \begin{cases} argmax_{u \in N_i}\{[\tau(i,u)][\eta^k(i,u)]^\beta\} & , \text{ if } q \leq q_0 \\ S & , \text{ otherwise} \end{cases} , \tag{6.7}$$

where N_i is the set of nodes associated to object i, j is the chosen cluster, $\tau(i,u)$ is the pheromone value between i and u, q_0 is the exploitation probability, q is a random number for strategy selection,β is a parameter, $\eta^k(i,u)$ is the heuristic value between i and u for ant k defined by the formula:

$$\eta^k(i,u) = 1/d(x_i, c_j^k) = ||x_i - c_j^k|| \ , \tag{6.8}$$

where x_i is a data instance and c_j^k is a centroid from the ant centroid list. and S is the exploration defined by:

$$S = P^k(i,u) = \frac{[\tau(i,u)][\eta^k(i,u)]^\beta}{\sum_{j=1}^{m}[\tau(i,j)][\eta^k(i,j)]^\beta} \ . \tag{6.9}$$

The algorithm steps can be divided in:

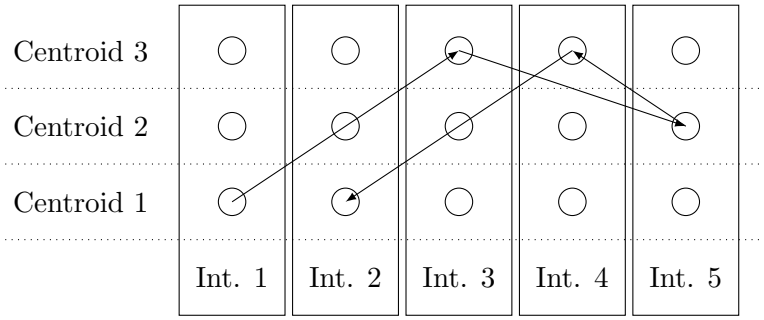


Figure 6.6: The graph construction in SACOC. The arrows represent a trail of an ant.

1. Initialize the pheromone matrix.
2. Initialize the ants: (tb^k, C^k, W^k), for each ant k in the colony. Then, each ant repeats until tb^k is full:
   (a) Select (randomly) a data object i satisfying $i \notin tb^k$.
   (b) Select a cluster j: first the ant chooses a strategy; then, it calculates the transition probability and, finally, it visits a node.
   (c) Update tb^k, C^k and W^k.
3. Choose the best solution. First, the objective function for each ant must be calculated as follows:
$$J^k = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}^k d(x_i, c_j^k) \ . \tag{6.10}$$
   where w_{ij}^k is a weight value of the assignation matrix W^k. Next, rank ants solutions. Choose the iteration-best solution, apply local search[1] to improve the solution and, finally, compare it with the best-so-far solution and update this value with the maximum between them.
4. Update the pheromone trails (global updating rule). Only the best r ants are able to add pheromones. Let ρ be the pheromone evaporation rate, $(0 < \rho < 1)$, t the iteration number, r is the number of elitism ants and $\Delta\tau_{ij}^h = 1/J^h$:
$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{h=1}^{r} w_{ij}^h \Delta\tau_{ij}^h \ . \tag{6.11}$$
5. Check the termination condition: if the number of iterations is greater than the maximum limit, finish; otherwise, go to step 2.

6.5.2 The Spectral hybridisation

The original ACOC algorithm uses the euclidean space as a search space. However, the algorithm can be modified to consider any kernel in a similar way that K-means is modified to generate the

[1] For more details of local search see [179].

Spectral Clustering algorithm. Consider a graph G and its associated weighted matrix W, which is a pairwise similarity graph amongst the data. The similarity is calculated using a similarity function defined by a kernel $k(x_i, x_j)$. The Spectrum of the graph is calculated in a similar way used by Ng et al. [155] to create the original Spectral Clustering algorithm. First, we calculate the Laplacian matrix defined by:

$$L = I - D^{-1/2} W D^{-1/2} \ , \tag{6.12}$$

where I is the identity matrix and D represents the diagonal matrix whose (i, i)-element is the sum of the similarity matrix i-th row. After the creation of the Laplacian matrix, we extract the $v_1, \ldots, v_z$, which corresponds with the z largest eigenvectors of L—chosen to be orthogonal to each other in the case of repeated eigenvalues—and form the matrix $V = [v_1\, v_2\, \ldots\, v_z] \in \mathbb{R}^{n \times z}$ by stacking the eigenvectors in columns. Finally, we form the matrix Y from V by renormalizing each row of V to have unit length (i.e., $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$). Then, we can consider Y as a projection of the original space and apply ACOC to the representation of each point.

6.6 Experiments

This section shows the experimental results. First, the synthetic and real-world datasets are briefly described. Then, the experimental setup is shown. Finally, the computational results for both synthetic and real-world datasets are discussed.

6.6.1 Datasets Description

As have been used in previous chapters to evaluate the performance of the different algorithms designed, for the synthetic experiments the next datasets [139] have been employed:

- *Aggregation*: This dataset is composed by 7 clusters, some of them can be separated by parametric clustering;
- *Jain*: This dataset is composed by two surfaces with different density and a clear separation;
- *Spiral*: In this case, there are 3 spirals close to each other.

For the real-world experiments, we have chosen three datasets from UCI Machine Learning Repository [70]:

- *Iris*: Contains 50 instances distributed over 3 classes, with 4 attributes each;
- *Haberman*: Contains 306 instances distributed over 2 classes, with 3 attributes each;
- *Breast Tissue* (Bre. Tis.): Contains 106 instances distributes over 6 classes, with 10 attributes each.

6.6.2 Experimental Setup

We have chosen K-means [126], Spectral Clustering (SC) [155] and ACOC [100] clustering algorithms to compare the results of SACOC. K-means is an iterative algorithm based on centroids. The goal of the algorithm is to find the best centroid positions. It involves two steps: first, the data to the closest centroid (cluster) is assigned, and second, the new position of the centroid as a centroid of the data which has been assigned to it, is calculated. SC generates a similarity graph and extracts its spectrum as a projective space in order to applied a simple clustering algorithm (in this case K-means) to the projective data.

The parameters of ACOC and SACOC are: the ants number has been fixed to 10, the elitism is 1, the exploitation probability is 0.0001, the initial pheromone values have been set to $1/m$—where m is the number of clusters, $\beta = 2.0$, $\rho = 0.1$, the local search probability is 0.001 and the maximum number of iterations is 1000. These values have been chosen according to the original ACOC paper [100].

All algorithms need the number of cluster as an initial parameter. The experiments have been carried out 50 times using the Euclidean distance as the metric, except for Spectral Clustering and SACOC which use the Radial Basis Function. The evaluation of the experiments has been focused on two different ideas: the synthetic datasets have been evaluated according to the cluster discrimination, and the algorithm performance when discriminates the original clusters; the real-world datasets have been evaluated using the accuracy rate, in order to check how close the algorithm is to real criteria.

6.6.3 Synthetic Experiments

Figure 6.7 presents the visual (best) results of the algorithms when applied to the synthetic datasets. Table 6.3 shows the accuracy results on the same datasets.

Aggregation results show that SACOC achieved the best results and outperforms all the other algorithms—SACOC results are statistically significantly better than SC ($p = 1.062 \times 10^{-8}$). K-means and ACOC usually have the worse results in this dataset. Figure 6.7 shows that these algorithms are not able to define clusters on the left, where the cluster boundaries are not clear.

Jain results show that both SC and SACOC are able to discriminate the clusters in all cases (see Table 6.3), both algorithms achieve the same results without statistically significant differences between them. K-means achieves stable results (the standard deviation is 0), while ACOC obtains the worst results.

Spirals shows that both SC and SACOC are able to define the clusters continuity—SACOC achieved the best and most stable results (0 standard deviation), which are statistically significantly better than SC ($p = 0.02225$). K-means and ACOC are not able to define the continuity of the data due to the use of the Euclidean space, see Figure 6.7.

The Genetic Algorithms obtain the best results compared with the rest of the algorithms, which means that there are some features which can be improved in SACOC.

Overall, the results for the synthetic datasets show that SACOC achieves best results, with statistically significant differences when compared to SC. In the next section we will compare

SACOC	Min	Max	Median	Mean	SD
Aggregation	**98.60**%	**99.62**%	**99.24**%	**99.28**%	▲ ± 0.0022
Jain	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Spirals	**100.0**%	**100.0**%	**100.0**%	**100.0**%	▲ ± 0.0000
SC	Min	Max	Median	Mean	SD
Aggregation	63.96%	99.37%	88.39%	90.30%	± 0.0716
Jain	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Spirals	35.26%	**100.0**%	**100.0**%	93.20%	± 0.1724
K-means	Min	Max	Median	Mean	SD
Aggregation	66.88%	88.07%	78.55%	77.93%	± 0.0495
Jain	78.28%	78.28%	78.28%	78.28%	± 0.0000
Spirals	33.97%	34.94%	34.29%	34.41%	± 0.0020
ACOC	Min	Max	Median	Mean	SD
Aggregation	62.18%	86.17%	77.73%	77.07%	± 0.0516
Jain	73.19%	76.68%	74.80%	74.97%	± 0.0067
Spirals	33.65%	36.54%	35.26%	35.14%	± 0.0063
GGC	Min	Max	Median	Mean	SD
Aggregation	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Jain	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Spirals	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
MOGGC	Min	Max	Median	Mean	SD
Aggregation	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Jain	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000
Spirals	**100.0**%	**100.0**%	**100.0**%	**100.0**%	± 0.0000

Table 6.3: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the synthetic datasets. The p-values for the Wilcoxon test applied to SACOC and SC results are: Aggregation ($p = 1.062 \times 10^{-8}$), Jain ($p = 0$) and Spiral ($p = 0.02225$)—statistical significant improvements are indicated by a ▲ symbol.

the algorithms in real-world datasets.

6.6.4 Real-world Experiments

Table 6.4 shows the results of the algorithms applied to real-world datasets from UCI Machine Learning repository [70].

Breast Tissue dataset is more a spectral-like dataset. The data is continuous and the clusters do not intersects in several parts. In this case, both SACOC and SC achieved good results—SACOC results are statistically significantly better than SC ($p = 7.689 \times 10^{-9}$). K-means and ACOC have problems in discriminating the clusters information.

In Haberman case, SACOC achieved the best results, however, SC achieves the highest

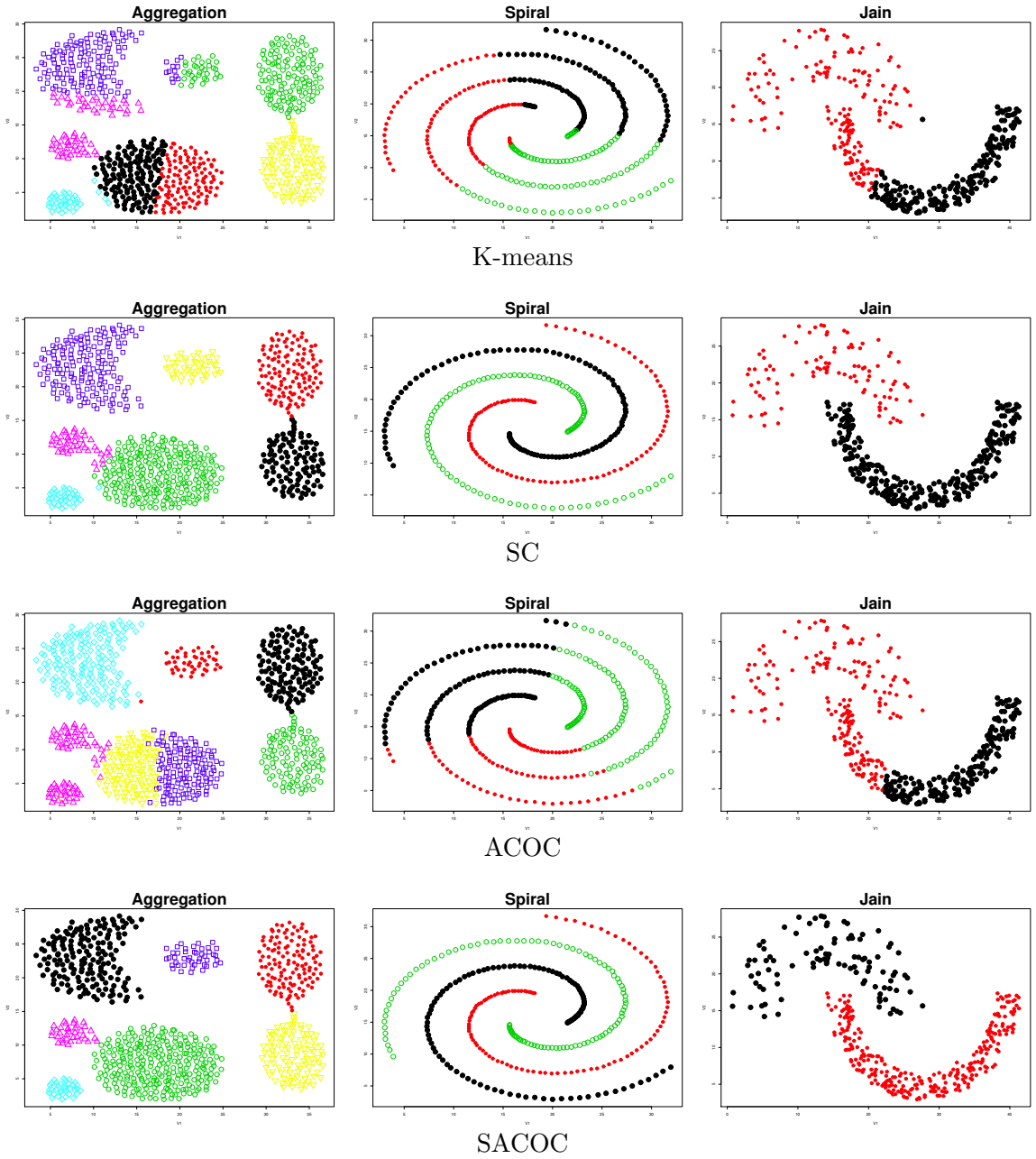


Figure 6.7: Graphical representation of the best results on the synthetic datasets.

maximum value. This datasets shows more stable results for SACOC than SC (the standard deviation of SACOC is 0). There is also a high statistical significance between them ($p = 3.919 \times 10^{-10}$). K-means and ACOC achieved the worse results again in this case.

Iris dataset shows interesting results. The best results are achieved by ACOC and K-means, while SACOC and SC achieved the worse results. This problem is likely due to the data projection, since it affects to both SC and SACOC. Usually, when there are places with cluster intersections, the data projection is generally –it worse produces a big cluster and a cluster with a couple of outliers. Even in this case, SACOC discriminates the clusters better than SC with statistically significant differences ($p = 5.371 \times 10^{-8}$).

SACOC	Min	Max	Median	Mean	SD
Breast Tissue	**39.62**%	**60.38**%	**48.11**%	**48.43**%	▲ ± 0.0432
Haberman	**73.53**%	73.53%	**73.53**%	**73.53**%	▲ ± 0.0000
Iris	66.67%	68.67%	68.67%	68.53%	▲ ± 0.0038
SC	Min	Max	Median	Mean	SD
Breast Tissue	36.79%	48.11%	41.51%	41.30%	± 0.0321
Haberman	51.31%	**75.82**%	52.12%	52.37%	± 0.0341
Iris	68.00%	68.00%	68.00%	68.00%	± 0.0000
K-means	Min	Max	Median	Mean	SD
Breast Tissue	33.02%	34.91%	33.02%	33.02%	± 0.0032
Haberman	50.00%	52.29%	51.96%	51.52%	± 0.0020
Iris	58.00%	89.33%	89.33%	84.95%	± 0.1098
ACOC	Min	Max	Median	Mean	SD
Breast Tissue	30.19%	40.57%	33.02%	33.42%	± 0.0184
Haberman	50.65%	52.94%	51.96%	51.90%	± 0.0048
Iris	**89.33**%	**92.67**%	**90.00**%	**90.23**%	± 0.0079
GGC	Min	Max	Median	Mean	SD
Breast Tissue	**60.38**%	**61.30**%	**60.38**%	*61.02*%	± 0.0101
Haberman	**74.21**%	**75.82**%	**74.52**%	**74.66**%	± 0.0109
Iris	**89.33**%	*92.00*%	*90.27*%	*90.31*%	± 0.0198
MOGGC	Min	Max	Median	Mean	SD
Breast Tissue	**60.38**%	**61.30**%	**60.38**%	**61.05**%	± 0.0112
Haberman	**74.21**%	**75.82**%	**74.52**%	*74.39*%	± 0.0107
Iris	**89.33**%	**96.00**%	**92.25**%	**92.22**%	± 0.0323

Table 6.4: Minimum, Maximum, Median, Mean and Standard Deviation accuracy results of the application of the algorithms to the synthetic datasets. The p-values for the Wilcoxon test applied to SACOC and SC results are: Breast Tissue ($p = 7.689 \times 10^{-9}$), Haberman ($p = 3.919 \times 10^{-10}$) and Iris ($p = 5.371 \times 10^{-8}$)—statistical significant improvements are indicated by a ▲ symbol.

The Genetic Algorithms obtain the best results compared with the rest of the algorithms, which means that there are some features which can be improved in SACOC, however, the algorithm shows interesting results compared with the UCI datasets.

These results show that SACOC achieved better and more stable results than SC in the datasets where the cluster assignation has clear boundaries and low cluster intersection. However, when there are intersection, it is harder for the algorithm to discriminate the data—in the same way that it is harder for SC.

6.7 Conclusions of SACOC

SACOC uses spectral transformations of the original search space in order to apply the clustering in the projective space. The transformation consists on converting the original data in a graph-based representation (through a similarity graph) and calculate its Laplacian matrix. Once the Laplacian has been obtained, the eigenvectors are extracted and normalized to generate the projective space.

The proposed SACOC algorithm showed good results for synthetic datasets. It is able to discriminate continuity-based clusters with more stable results, when compared to Spectral Clustering (SC). Also, the SACOC shows good results for real datasets, except in those cases where there are cluster intersections. In this situation, it has the same problems to discriminate the data than SC.

Chapter 7

Conclusions and Future Work

"L'avenir a plusieurs noms.
Pour les faibles, il se nomme l'impossible;
pour les timides, il se nomme l'inconnu;
pour les penseurs et pour les vaillants, il se nomme l'idéal."

- Victor Hugo

7.1 Conclusions

Spectral Clustering has been extensively used in clustering applications. This algorithm generates a graph topology on the data, but it has several problems according to its robustness when it discriminates the groups. Taking the same graph topology, the main question is whether it is possible to generate other clustering algorithms that could improve the current state of the art, oriented to cut the graph, through genetic algorithms. Also, it is important to determine different application fields and data structures to check its performance. Based on this idea, it is also important to analyse other bio-inspired approaches based on graph representations.

This dissertation has been divided in four main parts which cover the main contributions of the thesis. The main goal was the design of clustering algorithms which could combine graph clustering and genetic algorithms. We have developed three different generations of algorithms, which have been divided in three main parts:

- The first generation is focused on the design of GGC algorithm, a simple Genetic Graph-based Algorithm which uses a hybrid fitness to guide the search. This algorithm presents several memory problems (similar to Spectral Clustering) which have been analysed and solved in the next generation.

- The second generation is related to MOGA algorithms. In this generation we have implemented two different algorithms MOGGC and CEMOG. MOGGC is a multi-objective genetic algorithm designed to reduce the memory usage during the process, and CEMOG is an algorithm which uses a co-evolutionary approach combined with the previous multi-objective algorithm in order to generate a k-adaptive approach.

- The third generation deals with large and streaming data problems. In this case, the algorithms have been re-designed to process more data in limited computers. In order to achieve this goal, we have focused the problem on Map-Reduce and Online approaches creating two new algorithms: MOGGSC for Large Offline data, and MOGGOC for Stream Online data.

Other bio-inspired graph-based algorithms have been designed during the dissertation to study swarm-based approaches in the same context. These algorithms are based on Ant Colony Optimization algorithms, and have proved to provide promising results for future research.

With this characterization, it is possible to draw an answer to the main research question that was briefly proposed in the Introduction. Since the main research question is too wide and involves a collection of different issues, it was divided into seven specific research questions. In the following, we review these questions, which were already presented in the introduction, and their answers are discussed under the light of the experimental evidence reported along this dissertation.

- **Q1:** *Is it possible to improve the results of classical algorithms -specially Spectral Clustering, using genetic graph-based approaches?*

  Classical clustering algorithms have been focused on Gaussian models based on the optimization of the parameters of these models. K-means, which is the most classical algorithm, optimizes the position of centroids while Expectation Maximization (using a Gaussian Mixture Model) optimizes the mean and standard deviation of the distributions. These classical ideas are really popular and have been successfully applied in several fields, however, current clustering problems are not longer based on centroids, but manifolds.

  These manifolds (or special structures) are featured by the continuity defined by the data which compose them. This supposes the introduction of new algorithms based on data continuity. The most popular is Spectral Clustering.

  Some problems of the Spectral clustering algorithm are the robustness and the memory efficient of the algorithm.

  The first problem is related to how the algorithm deal with the data distribution in the metric space. If the distance between the points varies according to different magnitude orders, but the points are identically distributed than in the original space, then the algorithm may be less stable. This robustness problem is faced using the GGC algorithm. The continuity metric of this algorithm tries to improve the robustness of the solutions while keeps good results during the clustering process.

  The second problem is according to memory consumption. Spectral Clustering usually works with several matrices during the projection process, these matrices and the operations needed to generate it, usually take up a lot of memory specially when SC uses a full Similarity Graph. MOGGC has been designed to reduce this dependency. Its memory usage grows in a linear way while SC memory usage usually grows in a square way.

- **Q2:** *How these approaches deal with the robustness problem of classical Spectral Clustering algorithm?*

The genetic algorithms are designed using a specific encoding -which defines the search space, and the operations -that can be applied to guide the searching process, and the fitness -which provides information about how the current solutions found are close to the optimal solution. These aspects are important during the convergence process and are extremely important in order to solve the robustness problem keeping a good compromise with the convergence.

The way that genetic algorithms were chosen is also relevant. The first implementation, GGC, is a basic Genetic Algorithm. This version has provided good robustness results. However, it is not a good choice due to there are several aspects which are not well designed, such as the hybrid fitness function which should be a Multi-optimization problem. The second algorithm, MOGGC, is a Multi-Objective Genetic Algorithm, which improves the way the algorithm chooses the solution in the solution space defined by the two heuristics considered.

During the dissertation there are several metrics which are used in order to improve the algorithm results keeping their robustness. Also classical and promising metrics have been shown to be less useful in the field, e. g., the Weighted Clustering Coefficient metric.

The metrics which have shown to be helpful are those based on neighbourhood and continuity.

- **Q3:** *How can genetic algorithms and graph-based structures be combined to create these algorithms?*

One of the most important features of these algorithms (GGC, MOGGC and CEMOG) is to know how they can keep information about the graph, whereas the structure of the different chromosomes is reduced. It is important to note that the algorithms must have access to the graph which contains the search space. However, it is also important to look for a compromise between the information available and the memory usage. During the first approach, GGC used all the information of the structure, which means that it generates a $N \times N$ matrix for the full Similarity Graph. Taking into account the chromosome size (N) and the population size, this was really big.

Once we optimized the Similarity Graph using a $N \times K$ matrix, we simplified the problem, but we also reduced the information related to the search space. The most important feature of the new search space was whether it was able to provide enough information to find a good solution. That supposes to add some modifications to the way the algorithm search, adding two new fitness functions to it. These fitnesses show good performance, and also they provide a good solution to the algorithm. However, the chromosome size was still N. Also, the MOGGC extension, CEMOG, only add more space to the population but it helped to find the best k parameter.

Finally, the online and large data analysis part is focused on how to select relevant points of the search space in order to perform the resources usage (memory and CPU). Based on the idea that the continuity-based clusters are manifolds, and that a manifold is composed by local Euclidean spaces, we generate two steps during the search process. The first step looks for relevant local points of the manifold (considering this as a local space and generating Voronoi Regions). The second uses only the regions to define the final clusters. In this step we applied a graph topology to the regions in order to decide the best way to join them.

- **Q4:** *How modern genetic-based methodologies can help to improve the quality of the algorithms?*

  In this work three different approaches based on genetic algorithms have been used to improve the algorithms performance, as was mentioned above:

  - The first one is based on a simple genetic algorithm. This kind of algorithm helps to optimize the fitness function designed for GGC, looking for the best clustering assignation. The solutions provided by this algorithm are good enough, when they are compared against classical algorithms. However, the memory usage is too high. Also, due to the fitness defines two main objectives, the algorithm was re-designed as a multi-objective genetic algorithm.
  - The second approach -the multi-objective based; has provided several improvements in the algorithm. It achieves good results using less memory and less information about the search space, which is really helpful for the new implementations and for the large scale data analysis.
  - The last kind of genetic algorithm designed has been the Co-Evolution. This methodology was helpful in order to find a promising number of clusters. However, this process introduces some complexity in the genetic search, which was worse for the performance due to there were several populations sharing information between them. This reduces the scalability of the algorithm, which implies that is not a good choice for large scale data analysis.

- **Q5:** *How can these algorithms deal with static data?*

  As Chapters 3, 4, and 5 have shown, the new algorithms provide promising results for static datasets. They have been compared against classical algorithms, each of these algorithms is one of the most relevant in the field. In the first case, they have been compared with K-means, EM and Spectral Clustering. In order to guarantee the performance of the algorithms we have compared them using synthetic data and real-world datasets.

  The synthetic datasets have been designed to measure the algorithms behaviour in different contexts. The main context are: continuity-based clusters, centroid-based clusters and low noisy conditions. Using these contexts we are able to identify the different weak points of the algorithms in order to improve them in the next generation.

  The main weakness of the algorithms designed are related to the noise effects, and, specially, the cluster intersections.

  The main strengths of the algorithms are that they usually obtained better results than the classical algorithms, and they are able to deal with the continuity-based problem in a competitive way. According to the resources consumed, they reduce the memory usage in each generation providing more versatility for large scale data analysis.

- **Q6:** *How can these algorithms deal with streaming data?*

  The first and second generation of the algorithms need more improvements in order to deal with the streaming process. The main problem of these algorithms is not just the memory consumption problem, but also the problem of incorporate real-time data to the graph

structure. They are able to deal with the data provided by the streams offline, analysing a specific moment instead of the whole process.

The MOGGSC is also able to analyse a specific moment of the stream, but it is not able to analyse the process in a real-time context. MOGGOC instead is able to analyse the stream online, which means that it can provide the results each time they are needed. This is really useful for the different real-time data flows that can be found in real life (such as sensors, social network interactions, etc), however there are some limitations for this approach.

The main limitation for MOGGOC is that it can only deal with data which is contained in a search space, this means that the algorithm needs to know the search space in order to optimize the centroid positions during the first step, (this was not necessary in the previous generations). This is a common problem in all the online algorithms because the data stream usually provides data in a vectorial space way. However, Social Networks do not always provide data in this way, and some transformations are needed, in order to translate this information to a specific and known search space.

- **Q7:** *Is there any other bio-inspired methodology promising for graph-based approaches?*

We have seen that there are fields such as Swarm intelligence which can be also useful for clustering. The main idea behind the use of swarm intelligence is to try to define new bio-inspired algorithms that could be used to design new continuity clustering algorithms. This work was focused on one of the most relevant graph-based areas: Ant Colony Optimization (ACO). ACO algorithms have shown to have the potential to find more accurate solutions, to perform a deeper search in the search space and to avoid to trap in local minima. However, these algorithms where not so promising for continuity-based problems. The performance of ACO algorithms was specially focused on classical problems.

During the dissertation, we have developed a couple of ACO algorithms, the first was a medoid-based clustering algorithm (which was used to improve the convergence of the classical clustering ACO-based approach, and to extend this algorithm, named ACOC, to an unknown search space). The second was a spectral extension of the ACOC algorithm in order to create a continuity-based algorithm.

The results obtained from both algorithms in synthetic and real datasets have shown some interesting and promising results.

7.2 Future Work

There are several lines of work that could be extended in the near future related to the different algorithms and applications presented in this dissertation:

- The first issue is to provide a detailed computational convergence analysis for all the algorithms designed.
- It could be also important to test the behaviour of these algorithms in other application fields, such as text summarization for the second and third generation, and also for the ACO-based algorithms.

- Related to the fitness functions, it is important to provide a detailed study of the different ways which they usually define in the search space, in order to find a better way to improve the convergence of the algorithm and to alleviate the search process.
- For the online algorithm, it is interesting to find a way to provide an analytical process avoiding the features of the search space.
- Finally, it could be also interesting to study the current applications from Apache Spark approach in order to cover the current data analysis demands.

Chapter 8

Conclusiones y Trabajo Futuro

"L'avenir a plusieurs noms.
Pour les faibles, il se nomme l'impossible;
pour les timides, il se nomme l'inconnu;
pour les penseurs et pour les vaillants, il se nomme l'idéal."

- Victor Hugo

8.1 Conclusiones

El algoritmo de Clustering Espectral se ha utilizado en muchas aplicaciones basadas en clustering. Este algoritmo genera una topología de grafo en los datos, pero tiene varios problemas con respecto a su robustez cuando discrimina los grupos. Tomando la misma topología generada por el algoritmo, la cuestión principal es si es posible generar otros algoritmos de clustering capaces de mejorar el estado actual de esta técnica (orientados también a cortar el grafo), a través de algoritmos genéticos. También es importante determinar diferentes campos de aplicación para comprobar su rendimiento. En base a esta idea, también es importante analizar otro enfoque bio-inspirado basado en la representación de grafos.

Esta tesis se ha dividido en cuatro partes principales, que cubren las principales aportaciones de la tesis. El objetivo principal era el diseño de algoritmos de clustering combinando el clustering basado en grafos y algoritmos geneticos. Hemos desarrollado tres generaciones de algoritmos divididos en tres partes principales:

- La primera generación es sobre el algoritmo GGC, un Algoritmo Genético basado en Grafos sencillo que utiliza una función de fitness híbrida para guiar la búsqueda. Este algoritmo tiene varios problemas de memoria (similares a los de Clustering Espectral), que se resuelven en la siguiente generación.

- La segunda generación es de algoritmos Multi-Objetivo. En esta generación podemos incluir MOGGC, un algoritmo genético multiobjetivo diseñado para reducir el uso de memoria durante el proceso; y CEMOG, un algoritmo que utiliza un enfoque co-evolutivo, combinado con el algoritmo multi-objetivo anterior, con el fin de generar un algoritmo k-adaptativo.

- La tercera generación se enfoca en problemas de grandes cantidades de datos y streaming. En este caso, los algoritmos son re-diseñados para procesar más datos en ordenadores con capacidades limitadas. Para lograr este objetivo, nos hemos centrado en las metodologías de MapReduce y en enfoques Online, creando dos nuevos algoritmos: MOGGSC para flujos de datos offline y MOGGOC para los flujos online.

Después de estas generaciones también introducimos otros algoritmos bio-inspirados basados en grafos que se han diseñado durante la tesis. Estos algoritmos se basan en algoritmos de optimización de colonias de hormigas y han demostrado ofrecer resultados prometedores para investigaciones futuras. Con esta caracterización, es posible encontrar una respuesta a la pregunta principal de investigación. Dado que la pregunta de investigación es muy amplia e incluye una colección de diferentes cuestiones, era conveniente dividirlo en siete preguntas específicas. A continuación se revisan estas preguntas, que ya fueron presentadas en la introducción, y sus respuestas se discuten a la luz de las evidencias reportadas por esta tesis.

- **Q1:** *¿Es posible mejorar los resultados de los algoritmos clásicos -especialmente Clustering Espectral-, utilizando un enfoque de algóritmos genéticos basados en grafos?*

  Los algoritmos clásicos de clustering se han centrado en la optimización de parámetros de modelos típicamente Gaussianos. K-means, que es el algoritmo más clásico, optimiza la posición de los centroides mientras que Expectation Maximization (utilizando un modelo de múltiples gaussianas) optimiza la media y la desviación estándar de la distribución. Estas ideas clásicas son muy populares y se han aplicado con éxito en varios campos, sin embargo, los problemas actuales de clustering ya no se basan en centroides, sino en formas.

  Estas formas o estructuras especiales son caracterizadas por la continuidad que definen los datos que las componen. Esto supone la introducción del nuevas metodologías basadas en la continuidad de los datos. El algoritmo más importante actualmente es Clustering Espectral.

  Algunos de los problemas del algoritmo de Clustering Espectral son la robustez y la eficiencia en memoria del algoritmo.

  El primer problema está relacionado con los parámetros de la métrica que el algoritmo aplica a la distribución de datos en el espacio. Si la distancia entre los puntos varía de acuerdo con diferentes órdenes de magnitud, pero los puntos se distribuyen de manera idéntica que en el espacio original, entonces el algoritmo puede ser menos estable. Este problema de robustez se compara con el algoritmo GGC. La métrica de continuidad de este algoritmo intenta mejorar la robustez de las soluciones, mientras mantiene buenos resultados durante el proceso de agrupamiento.

  El segundo problema está enfocado a la eficiencia en memoria. Clustering Espectral generalmente trabaja con varias matrices durante el proceso de proyección, estas matrices y las operaciones necesarias para generarlo, por lo general, ocupan una gran cantidad de memoria, especialmente cuando SC utiliza un grafo de similitud completo. MOGGC ha sido diseñado para reducir esta dependencia. Su uso de la memoria crece de forma lineal mientras que el uso de memoria de SC suele crecer de forma cuadrática.

- **Q2:** *¿Cómo tratan estos enfoques el problema de la robustez del algoritmo de Clustering Espectral?*

Los algoritmos genéticos se diseñaron utilizando una codificación -que define el espacio de búsqueda-, las operaciones -que mueven la búsqueda-, y la fitness -que proporciona información sobre cómo las diferentes soluciones están cerca de la solución principal-. Estos aspectos son importantes durante el proceso de convergencia y son extremadamente importantes a la hora de resolver el problema de la robustez manteniendo un buen compromiso con la convergencia.

La forma en que se eligen los algoritmos genéticos también es relevante. La primera aplicación, GGC, es un algoritmo genético básico. Esta versión es una buena primera aproximación, sin embargo, no es una buena opción, debido a que hay varios aspectos que no están bien diseñados, tales como la función híbrida de fitness, que debería ser un problema multi-objetivo. El segundo algoritmo, MOGGC, es un algoritmo genético multi-objetivo, que mejora la forma en que el algoritmo elige la solución en el espacio de soluciones definidos por las dos heurísticas.

Durante la tesis hay varias métricas que se utilizan con el fin de mejorar los resultados del algoritmo manteniendo su robustez. También métricas clásicas y prometedoras han demostrado ser menos útiles en el campo, e. g., el coeficiente de clustering ponderado.

Las métricas que han demostrado ser útiles son aquellas basadas en vecinos y continuidad.

- **Q3:** *¿Cómo se pueden combinar los algoritmos genéticos y las estructuras basadas en grafos para generar estos algoritmos?*

Una de las partes más importantes del algoritmo es cómo se puede mantener la información sobre la estructura del grafo para los diferentes cromosomas. Es importante que los algoritmos tengan acceso al grafo que contiene el espacio de búsqueda. Sin embargo, también es importante tener un compromiso entre la información disponible y el uso de memoria. Durante el primer enfoque, GGC utiliza toda la información de la estructura, lo que significa que genera una matriz $N \times N$ para el grafo de similitud completo. Teniendo en cuenta el tamaño de los cromosomas (N) y el tamaño de la población, esto era realmente grande.

Una vez que hemos optimizado el grafo de similitud utilizando una matriz $N \times K$, no sólo hemos simplificado el problema, sino que también reducimos la información sobre el espacio de búsqueda. La característica más importante del nuevo espacio de búsqueda era si era capaz de proporcionar suficiente información para encontrar una buena solución. Eso supone añadir algunas modificaciones a la forma en que el algoritmo busca, añadiendo dos nuevas funciones de aptitud para ello. Estas funciones muestran un buen rendimiento, y también proporcionan una buena solución para el algoritmo. Sin embargo, el tamaño de los cromosomas era todavía N. Finalmente, la extensión de MOGGC, CEMOG, aunque agrega más espacio a la búsqueda, permite encontrar el parámetro k de forma automática.

Por último, la parte de análisis de flujos de datos, y grandes cantidades, se centra en cómo seleccionar puntos pertinentes al espacio de búsqueda con el fin de optimizar el uso de recursos (memoria y CPU). Basado en la idea de que los clusters basados en continuidad son variedades y que un centroide está asociado a un espacios euclídeo local, generamos un proceso de dos pasos durante la búsqueda. El primer paso busca puntos locales importantes en la variedad (considerando esto como un espacio local y generando regiones de Voronoi). El segundo utiliza sólo las regiones para definir los grupos finales. En este paso se aplicó una topología de grafo para las regiones con el fin de decidir la mejor manera de unirlas.

- **Q4:** *¿Cómo ayudan a mejorar la calidad de los algoritmos las técnicas modernas de algoritmos genéticos ?*

  En este trabajo se han utilizado tres enfoques diferentes basados en algoritmos genéticos para mejorar el rendimiento de los algoritmos, como se ha mencionado más arriba:

  - El primero es el algoritmo genético simple. Este algoritmo ayuda a optimizar la función de aptitud diseñada para GGC, buscando la mejor asignación de clustering. Las soluciones aportadas por este algoritmo son buenas, sin embargo, el uso de memoria es alto. También, debido a que la fitness define dos objetivos principales, el algoritmo evolucionó a un algoritmo genético multi-objetivo.
  - En la segunda generación se ha tratado de mejorar el rendimiento a través de algoritmos multi-objetivo. Se logran buenos resultados usando menos memoria y menos información sobre el espacio de búsqueda, lo cual, es realmente útil para las nuevas implementaciones y para el análisis de datos de gran tamaño.
  - El último tipo de algoritmo genético utilizado es el Co-Evolutivo. Esta metodología fue útil a la hora de encontrar un número óptimo de clusters. Sin embargo, este proceso introduce cierta complejidad en la búsqueda genética, que era peor para el rendimiento, debido a que hay varias poblaciones que comparten información entre ellas. Esto reduce la escalabilidad del algoritmo y no lo convierte en una buena elección para el análisis de datos de gran tamaño.

- **Q5:** *¿Cómo tratan estos algoritmos los datos estáticos?*

  Como los capítulos 3, 4 y 5 han mostrado, los nuevos algoritmos proporcionan resultados prometedores para datos estáticos. Éstos se han comparado con los algoritmos clásicos, cada uno de los algoritmos es uno de los algoritmos más relevantes utilizados en el campo. En el primer caso, se han comparado con K-means, EM y SC. Con el fin de garantizar el rendimiento de los algoritmos, se han utilizado datos sintéticos y conjuntos de datos del mundo real.

  Los datos sintéticos han sido diseñados para medir el comportamiento de los algoritmos en diferentes contextos. Los contextos principales son: clusters basados continuidad, clusters basados en centroides y condiciones de ruido. Aplicando estos contextos, podemos identificar los diferentes puntos débiles de los algoritmos con el fin de mejorarlos para la próxima generación.

  La principal debilidad de los algoritmos diseñados es que son sensibles a los efectos del ruido, y, especialmente, a las intersecciones entre clusters.

  Los principales puntos fuertes de los algoritmos es que, por lo general, obtienen mejores resultados que los algoritmos clásicos, y son capaces de lidiar con el problema de continuidad de una manera competitiva. Según los recursos consumidos, reducen el uso de memoria en cada generación proporcionando más versatilidad para el análisis de datos de gran tamaño.

- **Q6:** *¿Cómo tratan estos algoritmos los datos en streaming?*

  La primera y la segunda generación de algoritmos necesitan más mejoras con el fin de abordar el proceso de streaming. El principal problema de estos algoritmos no es sólo el

problema de la memoria que consumen, sino también el problema de incorporar datos en tiempo real a la estructura del grafo. Son capaces de hacer frente a los datos proporcionados por los flujos de datos de forma offline, pero no online.

El MOGGSC también es capaz de analizar un momento específico del flujo, pero no es capaz de analizar el proceso en un contexto en tiempo real. MOGGOC, sin embargo, es capaz de analizar los datos online, lo que significa que puede proporcionar los resultados cada vez que se necesiten. Esto es realmente útil para los diferentes flujos de datos en tiempo real que se pueden encontrar en el mundo (como sensores, redes sociales, etc), sin embargo, hay algunas limitaciones de este enfoque.

La principal limitación para MOGGOC es que sólo se puede hacer frente a los datos que se contienen en un espacio de búsqueda, lo que significa que tiene que saber el espacio de búsqueda con el fin de optimizar las posiciones del centroide durante la primera etapa, mientras que esto no era necesario en las generaciones anteriores. Este es un problema común en todos los algoritmos online, porque se suelen proporcionar flujos de datos sobre un espacio vectorial. Sin embargo, las redes sociales, por ejemplo, no siempre proporcionan datos de esta manera y hay que realizar algunas transformaciones con el fin de traducir esta información a un espacio de búsqueda específico y conocido.

- **Q7:** *¿Hay otros enfoques bio-inspirados prometedores para los problemas basados en el grafos?*

  Hemos visto que hay campos como la inteligencia de enjambre que pueden ser también útiles para el clustering. La idea principal detrás del uso de la inteligencia de enjambre es tratar de definir nuevos algoritmos bio-inspirados. Este trabajo se centra en una de las áreas basadas en grafos más relevantes: Ant Colony Optimization (ACO). Los algoritmos ACO han demostrado tener el potencial para encontrar soluciones más precisas, para llevar a cabo una búsqueda más profunda en el espacio de búsqueda y para evitar caer en mínimos locales. Sin embargo, estos algoritmos no son tan prometedores para los problemas de continuidad, aún. El rendimiento de los algoritmos ACO fue especialmente evaluado con problemas clásicos.

  Durante la disertación, hemos desarrollado un par de algoritmos ACO, el primero era un algoritmo basado en medoides (que se utiliza para mejorar la convergencia del problema clustering clásico y extender el algoritmo ACOC a un espacio de búsqueda desconocido). Y la segunda fue una extensión espectral del algoritmo ACOC el fin de crear un algoritmo basado en continuidad.

8.2 Trabajo Futuro

Hay varios temas que se pueden estudiar en el futuro de acuerdo a los diferentes algoritmos y aplicaciones de esta tesis:

- El primero es el de proporcionar una complejidad computacional detallada para todos los algoritmos.

- También es importante comprobar el comportamiento de los algoritmos en diferentes campos de aplicación, como el de resúmenes automáticos para la segunda y tercera generación y también para los algoritmos basados en ACO.
- Para las funciones de fitness, es importante proporcionar un estudio detallado de los diferentes caminos que, por lo general, se definen en el espacio de búsqueda con el fin de encontrar una manera de mejorar la convergencia del algoritmo y facilitar el proceso de búsqueda.
- Para el algoritmo online, es interesante encontrar una manera que proporcione un proceso analítico sin las características del espacio de búsqueda.
- También es interesante ampliar las aplicaciones para un enfoque Spark Apache con el fin de cubrir las demandas actuales de análisis de datos.

Bibliography

[1] Medical subject headings. http://www.nlm.nih.gov/mesh/.

[2] Metamap. http://metamap.nlm.nih.gov/.

[3] Systematized nomenclature of medicine - clinical terms. http://www.ihtsdo.org/snomed-ct/.

[4] Unified medical language system. http://www.nlm.nih.gov/research/umls/.

[5] *Advanced Lectures on Machine Learning*, volume 33. Springer, 2012.

[6] Biomed central corpus, 2012. http://www.biomedcentral.com/about/datamining.

[7] K. Adamska. Cluster analysis of genetic algorithms results. *Inteligencia Artificial, Revista Iberoamericana de IA*, 9(28):25–32, 2005.

[8] S. Afantenos, V. Karkaletsis, and P. Stamatopoulos. Summarization from medical documents: a survey. *Artificial Intelligence in Medicine*, 33(2):157–177, 2005.

[9] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer, 2007.

[10] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[11] J. Aguilar. Resolution of the clustering problem using genetic algorithms. *International Journal of Computers*, 1(4):237 – 244, 2007.

[12] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503 – 527, 2007.

[13] L. Araujo. Symbiosis of evolutionary techniques and statistical natural language processing. *Evolutionary Computation, IEEE Transactions on*, 8(1):14–27, 2004.

[14] L. Ashok and D. W. Messinger. A spectral image clustering algorithm based on ant colony optimization. volume 8390, pages 1–10, 2012.

[15] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[16] F. Bach and M. Jordan. Learning Spectral Clustering, With Application To Speech Separation. *Journal of Machine Learning Research*, 7:1963 – 2001, Oct. 2006.

[17] A. Banerjee. An improved genetic algorithm for robust fuzzy clustering with unknown number of clusters. In *2010 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 1–6. IEEE, July 2010.

[18] W. Barbakh and C. Fyfe. Clustering with reinforcement learning. In H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, volume 4881 of *Lecture Notes in Computer Science*, pages 507–516. Springer Berlin / Heidelberg, 2007.

[19] W. Barbakh and C. Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(03):185–194, 2008.

[20] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, March 2004.

[21] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17, 1997.

[22] G. Bello, H. Menéndez, and D. Camacho. Using the clustering coefficient to guide a genetic-based communities finding algorithm. In H. Yin, W. Wang, and V. Rayward-Smith, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 of *Lecture Notes in Computer Science*, pages 160–169. Springer Berlin / Heidelberg, 2011.

[23] G. Bello-Orgaz, H. Menéndez, S. Okazaki, and D. Camacho. Combining social-based data mining techniques to extract collective trends from twitter. *Malaysian Journal of Computer Science*, 27(2), 2014.

[24] G. Bello-Orgaz, H. D. Menéndez, and D. Camacho. Adaptive k-means algorithm for overlapped graph clustering. *International Journal of Neural Systems*, 22(05):1250018, 2012.

[25] G. Bello-Orgaz, M. D. R-Moreno, D. Camacho, and D. F. Barrero. Clustering avatars behaviours from virtual worlds interactions. In *Proceedings of the 4th International Workshop on Web Intelligence; Communities*, pages 4:1–4:7, New York, NY, USA, 2012. ACM.

[26] M. W. Berry and M. Castellanos. Survey of text mining. *Computing Reviews*, 45(9):548, 2004.

[27] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Springer, 1 edition, Mar. 2005.

[28] P. Bientinesi, I. S. Dhillon, and R. A. Van De Geijn. A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations. *SIAM Journal on Scientific Computing*, 27(1):43–66, 2005.

[29] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 11:1–8, New York, NY, USA, 2004. ACM.

[30] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97:245–271, December 1997.

[31] S. P. Borgatti. Centrality and network flow. *Social Networks*, 27:55–71, 2005.

[32] R. Brandow, K. Mitze, and L. Rau. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 5(31):675–685, 1995.

[33] L. Bungum and B. Gamback. Evolutionary algorithms in natural language processing. In *Proceedings of the Norwegian Artificial Intelligence Symposium*, pages 7–19, 2010.

[34] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[35] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, volume 6, pages 326–337. SIAM, 2006.

[36] S. R. Carroll and D. J. Carroll. *Statistics Made Simple for School Leaders*. Rowman & Littlefield, 2002.

[37] H. Chang and D.-Y. Yeung. Robust path-based spectral clustering. *Pattern Recogn.*, 41(1):191–203, Jan. 2008.

[38] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, 2011.

[39] C. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

[40] K. J. Cios, R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan. Unsupervised learning: Clustering. In *Data Mining*, pages 257–288. Springer US, 2007.

[41] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111–1 – 066111–6, Dec. 2004.

[42] C. Coello, G. Lamont, and D. Van Veldhuisen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and evolutionary computation series. Springer Science+Business Media, LLC, 2007.

[43] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *J. Artif. Int. Res.*, 10(1):243–270, May 1999.

[44] R. M. Cole. Clustering with Genetic Algorithms. Master's thesis, Nedlands 6907, Australia, 1998.

[45] Coley. *An Introduction to Genetic Algorithms for scientists and engineers*. World Scientific Publishing, 1999.

[46] D. Corne, J. D. Knowles, and M. J. Oates. The pareto envelope-based selection algorithm for multi-objective optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, 2000. Springer-Verlag.

[47] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.*, 47(4):547–553, Nov. 2009.

[48] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 –27, january 1967.

[49] L. Curiel, B. Baruque, C. Dueñas, E. Corchado, and C. Pérez-Tárrago. Genetic algorithms to simplify prognosis of endocarditis. In *Proceedings of the 12th international conference on Intelligent data engineering and automated learning*, IDEAL'11, pages 454–462, Berlin, Heidelberg, 2011. Springer-Verlag.

[50] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):218 –237, jan. 2008.

[51] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[52] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 849–858, London, UK, 2000. Springer-Verlag.

[53] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1 edition, June 2001.

[54] M. Dehmer, editor. *Structural Analysis of Complex Networks*. Birkhäuser Publishing, 2010.

[55] K. Delac, M. Grgic, and S. Grgic. Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, 15(5):252–260, 2005.

[56] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[57] S. Ding. Feature selection based f-score and aco algorithm in support vector machine. In *Knowledge Acquisition and Modeling, 2009. KAM '09. Second International Symposium on*, volume 1, pages 19–23, Nov 2009.

[58] D. Doval, S. Mancoridis, and B. S. Mitchell. Automatic Clustering of Software Systems using a Genetic Algorithm. In *IEEE Proceedings of the 1999 Int. Conf. on Software Tools and Engineering Practice (STEP'99)*, pages 73–91, 1999.

[59] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[60] H. P. Edmundson. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, 2(16):264–285, 1969.

[61] A. E. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 582–587. IEEE, 2002.

[62] G. Erkan and D. R. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22:457–479, 2004.

[63] C. Fellbaum. *WordNet*. Wiley Online Library, 1998.

[64] X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 36:1–8, New York, NY, USA, 2004. ACM.

[65] V. Fernandez, R. G. Martinez, R. Gonzalez, and L. Rodriguez. Genetic algorithms applied to clustering. In *In Proceedings of the Winter Simulation Conference*, pages 1307–1314, 1997.

[66] M. Fiszman, T. C. Rindflesch, and H. Kilicoglu. Abstraction summarization for managing the biomedical research literature. In *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*, pages 76–83, 2004.

[67] M. Fiszman, T. C. Rindflesch, and H. Kilicoglu. Summarizing drug information in medline citations. In *Proceedings of AMIA Annu Symp*, pages 254–258, 2006.

[68] S. Fortunato, V. Latora, and M. Marchiori. Method to find community structures based on information centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056104–1–056104–13, 2004.

[69] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, 2004.

[70] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[71] A. A. Freitas. A review of evolutionary algorithms for data mining. In *Soft Computing for Knowledge Discovery and Data Mining*, pages 61–93, 2007.

[72] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An implementation of the look-ahead lanczos algorithm for non-hermitian matrices. *SIAM Journal on Scientific Computing*, 14(1):137–158, 1993.

[73] L. Fu and E. Medico. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinformatics*, 8:1–15, 2007.

[74] C. Fyfe. Topographic maps for clustering and data visualization. In J. Fulcher and L. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 111–153. Springer Berlin - Heidelberg, 2008.

[75] C. Fyfe and W. Barbakh. Immediate reward reinforcement learning for clustering and topology preserving mappings. In M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors, *Similarity-Based Clustering*, volume 5400 of *Lecture Notes in Computer Science*, pages 35–51. Springer Berlin - Heidelberg, 2009.

[76] S. Geisser. *Predictive Inference: An Introduction*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1993.

[77] S. Ghosh-Dastidar, H. Adeli, and N. Dadmehr. Principal component analysis-enhanced cosine radial basis function neural network for robust epilepsy and seizure detection. *Biomedical Engineering, IEEE Transactions on*, 55(2):512 –518, feb. 2008.

[78] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1):1–30, Mar. 2007.

[79] A. N. Gorban and A. Zinovyev. Principal manifolds and graphs in practice: From molecular biology to dynamical systems. *International Journal of Neural Systems*, 20(3):219 – 232, 2010.

[80] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pages 359–366. IEEE, 2000.

[81] M. R. Gupta and Y. Chen. Theory and use of the em algorithm. *Foundations and Trends in Signal Processing*, 4(3):223–296, 2010.

[82] P. Haider, L. Chiarandini, and U. Brefeld. Discriminative clustering for market segmentation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 417–425, New York, NY, USA, 2012. ACM.

[83] D. Hall and D. Klein. Finding cognate groups using phylogenies. In *Proceedings of the 48th Annual Meeting of the Association for Computationa Linguistics*, pages 1030–1039, 2010.

[84] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[85] J. Handl and J. Knowles. Evolutionary multiobjective clustering. In *In Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer, 2004.

[86] J. Handl and J. Knowles. Exploiting the Trade-Off - The Benefits of Multiple Objectives in Data Clustering. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 547–560, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[87] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4–6):175–181, 2000.

[88] N. P. Holden and A. A. Freitas. A hybrid pso/aco algorithm for classification. In *Proceedings of GECCO'07*, pages 2745–2750, New York, NY, USA, 2007. ACM.

[89] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[90] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho. A survey of evolutionary algorithms for clustering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(2):133 –155, march 2009.

[91] J. Hsu, R. Bravo, and R. Taub. Interactions among lrf-1, junb, c-jun, and c-fos define a regulatory program in the g1 phase of liver regeneration. *Mol Cell Biol*, 12(10):4654–4665, 1992.

[92] S. M. Humphrey, W. J. Rogers, H. Kilicoglu, D. Demner-Fushman, and T. C. Rindflesch. Word sense disambiguation by selecting the best semantic type based on journal descriptor indexing: Preliminary experiment. *J. Am. Soc. Inf. Sci. Technol.*, 57(1):96–113, Jan. 2006.

[93] B. Humphreys and D. McCutcheon. Growth patterns in the national library of medicine's serials collection and its index medicus journals, 1966-1985. *Bulletin of the Medical Library Association*, 82(1):18–24, 1994.

[94] C. Huttenhower, A. Flamholz, J. Landis, S. Sahi, C. Myers, K. Olszewski, M. Hibbs, N. Siemers, O. Troyanskaya, and H. Coller. Nearest Neighbor Networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, 8(1):250, 2007.

[95] O. M. Jafar and R. Sivakumar. Ant-based clustering algorithms: A brief survey. *International Journal of Computer Theory and Engineering*, 2:787–796, 2010.

[96] A. Jain and M. Law. Data clustering: A user's dilemma. In S. Pal, S. Bandyopadhyay, and S. Biswas, editors, *Pattern Recognition and Machine Intelligence*, volume 3776 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin / Heidelberg, 2005.

[97] G. Jiménez-Díaz, H. D. Menéndez, D. Camacho, and P. A. González-Calero. Predicting performance in team games. In I. I. for systems, C. Technologies of Information, and Communication, editors, *ICAART 2011 - Proceedings of the 3ed International Conference on Agents and Artificial Intelligence*, volume 1, pages pages 401 – 406, 2011.

[98] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[99] K. S. Jones and J. R. Galliers. *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[100] Y. Kao and K. Cheng. An aco-based clustering algorithm. In *Ant Colony Optimization and Swarm Intelligence*, volume 4150 of *Lecture Notes in Computer Science*, pages 340–347. Springer Berlin Heidelberg, 2006.

[101] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[102] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. 1987.

[103] K. Kim, R. B. McKay, and B.-R. Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1179–1186, New York, NY, USA, 2010. ACM.

[104] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[105] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, December 1997.

[106] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, 29(2):249–272, 2011.

[107] K. Krishna and M. N. Murty. Genetic K-means Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 29(3):433–439, 1999.

[108] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the 18th ACM/SIGIR Annual Conference on Research and Development in Information Retrieval*, pages 68–73, 1995.

[109] G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380, Feb. 1967.

[110] W. Langdon and R. Poli. Evolving problems to learn about particle swarm and other optimisers. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 81–88, sept. 2005.

[111] D. T. Larose. *Discovering Knowledge in Data*. John Wiley & Sons, 2005.

[112] J. Lee. *Introduction to smooth manifolds*, volume 218. Springer, 2012.

[113] J. Lee, L. Choi, and S. Park. Multi-objective genetic algorithms, nsga-ii and spea2, for document clustering. In T.-h. Kim, H. Adeli, H.-k. Kim, H.-j. Kang, K. Kim, A. Kiumi, and B.-H. Kang, editors, *Software Engineering, Business Continuity, and Education*, volume 257 of *Communications in Computer and Information Science*, pages 219–227. Springer Berlin Heidelberg, 2011.

[114] Y. Li, J. Chen, R. Liu, and J. Wu. A spectral clustering-based adaptive hybrid multi-objective harmony search algorithm for community detection. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[115] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451 – 461, 2003.

[116] C.-Y. Lin. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough. In *Proceedings of the NTCIR Workshop*, volume 4, pages 1–10, 2004.

[117] C. Y. Lin. Rouge: A Package for Automatic Evaluation of Summaries. In M. F. Moens and S. Szpakowicz, editors, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics.

[118] X. Ling, J. Jiang, X. He, Q. Mei, C. Zhai, and B. Schatz. Generating gene summaries from biomedical literature: A study of semi-structured summarization. *Inf. Process. Manage.*, 43(6):1777–1791, 2007.

[119] M. Lipczak and E. Milios. Agglomerative genetic algorithm for clustering in social networks. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1243–1250, New York, NY, USA, 2009. ACM.

[120] M. Litvak and M. Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[121] M. Litvak, M. Last, and M. Friedman. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 927–936, 2010.

[122] T. Y. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, Mar. 2009.

[123] H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2(2):159–165, 1958.

[124] S. Luke. Is the perfect the enemy of the good. In *In Genetic and Evolutionary Computation Conference*, pages 820–828. Morgan Kaufmann, 2002.

[125] D. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[126] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[127] I. Mani. Summarization evaluation: An overview. In *Proceedings of the 2nd NTCIR workshop on research in Chinese and Japonese text retrieval and text summarization. Tokio, Japan: National Institute of Informatics*, 2001.

[128] I. Mani and M. Maybury. *Advances in automatic text summarization.* The MIT Press, 1999.

[129] D. Martens, B. Baesens, and T. Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1–42, 2011.

[130] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[131] N. Matake, T. Hiroyasu, M. Miki, and T. Senda. Multiobjective clustering with automatic k-determination for large-scale data. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, page 861, New York, New York, USA, July 2007. ACM Press.

[132] U. Maulik. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.

[133] H. Menéndez, D. F. Barrero, and D. Camacho. A multi-objective genetic graph-based clustering algorithm with memory optimization. In *2013 IEEE Conference on Evolutionary Computation*, volume 1, pages 3174–3181, June 20-23 2013.

[134] H. Menendez, G. Bello-Orgaz, and D. Camacho. Features selection from high-dimensional web data using clustering analysis. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, pages 20:1–20:9, New York, NY, USA, 2012. ACM.

[135] H. Menéndez, G. Bello-Orgaz, and D. Camacho. Extracting behavioural models from 2010 fifa world cup. *Journal of Systems Science and Complexity*, 26(1):43–61, 2013.

[136] H. Menéndez and D. Camacho. A multi-objective graph-based genetic algorithm for image segmentation. In *Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014 IEEE International Symposium on*, pages 234–241. IEEE, 2014.

[137] H. Menéndez, F. E. B. Otero, and D. Camacho. Macoc: a medoid-based aco clustering algorithm. In *Swarm Intelligence - 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings*, pages 122–133. Springer, 2014.

[138] H. D. Menéndez, D. F. Barrero, and D. Camacho. A co-evolutionary multi-objective approach for a k-adaptive graph-based clustering algorithm. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2724–2731. IEEE, 2014.

[139] H. D. Menéndez, D. F. Barrero, and D. Camacho. A genetic graph-based approach for partitional clustering. *Int. J. Neural Syst.*, 24(3), 2014.

[140] H. D. Menéndez and D. Camacho. A genetic graph-based clustering algorithm. In H. Yin, J. Costa, and G. Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, volume 7435 of *Lecture Notes in Computer Science*, pages 216–225. Springer Berlin / Heidelberg, 2012.

[141] H. D. Menéndez, C. Delgado-Calle, and D. Camacho. Tweetsemminer: A meta-topic identification model for twitter using semantic analysis. In E. C. et al., editor, *Intelligent Data Engineering and Automated Learning - IDEAL 2014*, volume 8669 of *Lecture Notes in Computer Science*, pages 69–76. Springer International Publishing Switzerland, 2014.

[142] H. D. Menéndez, F. E. Otero, and D. Camacho. Sacoc: A spectral-based aco clustering algorithm. In *Intelligent Distributed Computing VIII*, pages 185–194. Springer, 2015.

[143] H. D. Menéndez, L. Plaza, and D. Camacho. A genetic graph-based clustering approach to biomedical summarization. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*, WIMS '13, pages 10:1–10:8, New York, NY, USA, 2013. ACM.

[144] H. D. Menéndez, L. Plaza, and D. Camacho. Combining graph connectivity and genetic clustering to improve biomedical summarization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2740–2747. IEEE, 2014.

[145] H. D. Menéndez, M. Vázquez, and D. Camacho. Mixed clustering methods to forecast baseball trends. In *Intelligent Distributed Computing VIII*, pages 175–184. Springer, 2015.

[146] H. D. Menéndez, R. Vindel, and D. Camacho. Combining time series and clustering to extract gamer profile evolution. In *Computational Collective Intelligence. Technologies and Applications - 6th International Conference, ICCCI 2014, Seoul, Korea, September 24-26, 2014. Proceedings*, pages 262–271. 2014.

[147] O. d. Merle, P. Hansen, B. Jaumard, and N. Mladenovic. An interior point algorithm for minimum sum-of-squares clustering. *SIAM J. Sci. Comput.*, 21(4):1485–1505, Dec. 1999.

[148] R. Mihalcea and P. Tarau. TextRank - Bringing order into text. In *Proceedings of the Conference EMNLP 2004*, pages 404–411, 2004.

[149] A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. Clustering using multi-objective genetic algorithm and its application to image segmentation. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 3, pages 2678–2683, Oct.

[150] B. Nadler, S. Lafon, R. Coifman, and I. G. Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. *Advance in Neural Information Processing Systems*, 18:955 – 962, 2006.

[151] M. C. V. Nascimento and A. C. P. L. F. Carvalho. A graph clustering algorithm based on a clustering coefficient for weighted graphs. *J. Braz. Comp. Soc.*, 17(1):19–29, 2011.

[152] G. Nathiya, S. C. Punitha, and M. Punithavalli. An analytical study on behavior of clusters using k means, em and k* means algorithm. *CoRR*, abs/1004.1743, 2010.

[153] S. Nelson, T. Powell, and B. Humphreys. The unified medical language system (umls) project. *Encyclopedia of library and information science.*, pages 368–378, 2002.

[154] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.

[155] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

[156] G. B. Orgaz, H. D. Menéndez, S. Okazaki, and D. Camacho. Extracting collective trends from twitter using social-based data mining. In *ICCCI*, pages 622–630, 2013.

[157] F. Otero, A. Freitas, and C. Johnson. Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, 12(11):3615–3626, 2012.

[158] F. Otero, A. Freitas, and C. Johnson. A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):64–76, 2013.

[159] A. Pascual, M. Barcéna, J. Merelo, and J.-M. Carazo. Application of the fuzzy kohonen clustering network to biological macromolecules images classification. In J. Mira and J. Sánchez-Andrés, editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks*, volume 1607 of *Lecture Notes in Computer Science*, pages 331–340. Springer Berlin Heidelberg, 1999.

[160] E. Pitler and A. Nenkova. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

[161] L. Plaza, A. Díaz, and P. Gervás. Concept-graph based biomedical automatic summarization using ontologies. In *TextGraphs '08: Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 53–56, 2008.

[162] L. Plaza, A. Díaz, and P. Gervás. A semantic graph-based approach to biomedical summarisation. *Artif. Intell. Med.*, 53(1):1–14, Sept. 2011.

[163] P. Pokorný and P. Dostál. Cluster analysis and genetic algorithms. In *In: Management, Economics and Business Development in the New European Conditions*, pages 1–9, 2008.

[164] R. Poli and W. B. Langdon. Backward-chaining evolutionary algorithms. *Artificial Intelligence*, 170(11):953 – 982, 2006.

[165] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, Dec. 2005.

[166] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[167] D. R. Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, H. Qi, A. Çelebi, D. Liu, and E. Drabek. Evaluation challenges in large-scale document summarization. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 375–382, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[168] L. Reeve, H. Han, and A. Brooks. The use of domain-specific concepts in biomedical text summarization. *Information Processing and Management*, 43:1765–1776, 2007.

[169] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.

[170] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli. Sortnet: Learning to rank by a neural-based sorting algorithm. In *In proceedings of the SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR)*, volume 42, pages 76–79, 2008.

[171] T. Rindflesch and M. Fiszman. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36:462–477, 2003.

[172] K. Ripon and S. Kwong. Multi-Objective Data Clustering using Variable-Length Real Jumping Genes Genetic Algorithm and Local Search Method. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3609–3616. IEEE, 2006.

[173] L. Rodríguez, I. García Varea, and J. Gámez. On the application of different evolutionary algorithms to the alignment problem in statistical machine translation. *Neurocomputing*, 71(4-6):755–765, 2008.

[174] V. Roth and T. Lange. Feature selection in clustering problems. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[175] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag, Heidelberg, New York, 2nd editio edition, 2006.

[176] D. K. Roy and L. K. sharma. Genetic kmeans clustering algorithm for mixed numeric and categorial data sets. *International Journal of Artificial intelligence and Applications(IJAIA)*, 1(2):23 – 28, 2010.

[177] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[178] Y. Shang, Y. Li, H. Lin, and Z. Yang. Enhancing biomedical text summarization using semantic relation extraction. *PLoS one*, 6(8):1–10, 2011.

[179] P. Shelokar, V. K. Jayaraman, and B. D. Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, 2004.

[180] L.-D. Shi, Y.-H. Shi, Y. Gao, L. Shang, and Y.-B. Yang. Xcsc:: A novel approach to clustering with extended classifier system. *International Journal of Neural Systems*, 21(1):79 – 93, 2011.

[181] Z. Shi, G. Melli, Y. Wang, Y. Liu, B. Gu, M. M. Kashani, A. Sarkar, and F. Popowich. Question answering summarization of multiple biomedical documents. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 284–295, 2007.

[182] Z. Shi, G. Melli, Y. Wang, Y. Liu, B. Gu, M. M. Kashani, A. Sarkar, and F. Popowich. Question answering summarization of multiple biomedical documents. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 284–295, 2007.

[183] M. R. Shintaro Okazaki, Ana M. Díaz-Martín and H. D. Menéndez-Benito. How to mine brand tweets: Procedural guidelines and pretest. *International Journal of Market Research*, 2014.

[184] M. R. Shintaro Okazaki, Ana M. Díaz-Martín and H. D. Menéndez-Benito. Using twitter to engage with customers: A data mining approach. *Internet Research*, 2014.

[185] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[186] T. Smith and I. Witten. Learning language using genetic algorithms. *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, vol. 1040 of LNAI*, pages 132–145, 1995.

[187] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656 –667, apr 1994.

[188] S. Tratz and E. Hovy. Summarization evaluation using transformed basic elements. In *Proceedings of the 1st Text Analysis Conference*, 2008.

[189] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: A ranking method with fidelity loss. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 383–390, New York, NY, USA, 2007. ACM.

[190] L. Y. Tseng and S. B. Yang. A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415 – 424, 2001.

[191] A. Tsonis, K. Swanson, and G. Wang. Estimating the clustering coefficient in scale-free networks on lattices with local spatial correlation structure. *Physica A: Statistical Mechanics and its Applications*, 387(21):5287–5294, 2008.

[192] C. Veenman, M. Reinders, and E. Backer. A maximum variance cluster algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(9):1273 – 1280, sep 2002.

[193] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007.

[194] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, Apr. 2008.

[195] M. D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA, 1998.

[196] H. Wang, J. Chen, and K. Guo. A genetic spectral clustering algorithm. *Journal of Computational Information Systems*, 7(9):3245–3252, 2011.

[197] L. Wang, M. Jiang, Y. Lu, M. Sun, and F. Noe. A comparative study of clustering methods for molecular data. *International Journal of Neural Systems*, 17(6):447 – 458, 2007.

[198] X. Wang, B. Qian, J. Ye, and I. Davidson. Multi-objective multi-view spectral clustering via pareto optimization. pages 234–242. SDM, 2013.

[199] T. Warren Liao. Clustering of time series data – a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[200] D. J. Watts. *Small worlds: The dynamics of networks between order and randomness*. Princeton University Press, Princeton, NJ, 1999.

[201] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, 1945.

[202] Wojciech and Kwedlo. A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, 32(12):1613 – 1621, 2011.

[203] R. Xu and D. Wunsch. *Clustering (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press, illustrated edition edition, Oct. 2008.

[204] S. Ye, T.-S. Chua, M.-Y. Kan, and L. Qiu. Document concept lattice for text understanding and summarization. *Information Processing and Management*, 43(6):1643–1662, 2007.

[205] I. Yoo and X. Hu. A comprehensive comparison study of document clustering for a biomedical digital library medline. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '06, pages 220–229, New York, NY, USA, 2006. ACM.

[206] I. Yoo, X. Hu, and I.-Y. Song. A Coherent Graph-Based Semantic Clustering and Summarization Approach for Biomedical Literature and a New Summarization Evaluation Method. *BMC Bioinformatics*, 8(9):S4, 2007.

[207] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 10:1–7, 2010.

[208] C. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, C-20(1):68 – 86, jan. 1971.

[209] H. Zhang, J. E. Fritts, and S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *computer vision and image understanding*, 110(2):260–280, 2008.

[210] X. Zhang, X. Chen, and Z. He. An aco-based algorithm for parameter optimization of support vector machines. *Expert Syst. Appl.*, 37(9):6618–6628, Sept. 2010.

[211] W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on mapreduce. In *Cloud Computing*, pages 674–679. Springer, 2009.

[212] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[213] P. Zweigenbaum, D. Demner-Fushman, H. Yu, and K. B. Cohen. Frontiers of biomedical text mining: current progress. *Briefings in Bioinformatics*, 8(5):358–375, 2007.